UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE INFORMÁTICA


# Carteira Digital Como Solução Financeira

Marcus Sousa Dias

**Mestrado em Engenharia Informática**
Especialização em Engenharia de Software

Trabalho de Projeto orientado por:
Prof. Doutora Maria Antónia Bacelar da Costa Lopes

2019

# Agradecimentos

Agradeço à minha orientadora, a Professora Doutora Maria Antónia Bacelar da Costa Lopes, que ao longo deste mestrado, para além de conhecimentos e ideias, inspirou-me a ser uma melhor pessoa e um melhor profissional. Agradeço pela paciência e tempo disponibilizado durante esta etapa da minha vida.

Agradeço à Innovation Makers e toda a equipa que durante todos os anos que estivemos juntos e principalmente durante o meu projeto de mestrado, demostraram ser como uma família, que me deu toda a ajuda necessária e suporte para ultrapassar as dificuldades encontradas. Agradeço especialmente ao meu supervisor, Francisco Van Uden Chaves por ter-me ajudado a ver o percurso a seguir, aquando em momentos, eu não sabia por onde trilhar e pela oportunidade que me proporcionou de poder realizar o projeto nesta instituição.

Agradeço a minha mãe, meu irmão e a Diana Ivakina, que me proporcionaram a oportunidade de iniciar meu percurso académico e que estiveram do meu lado durante todos os momentos difíceis, a apoiarem, a darem-me força e coragem para continuar. Sem a vossa presença permanente, este trabalho não seria possível.

Agradeço aos meus amigos, principalmente ao Daniel Santos, Mariana Amaral e ao João Calado por serem fontes inesgotáveis de inspiração, paciência e amizade. Obrigado pelo interesse constante e por acreditarem em mim.

# Resumo

A inclusão financeira tem vindo a ser cada vez mais um tema debatido na nossa sociedade e é reconhecido como um objetivo importante, cuja prossecução poderá impulsionar a economia. É, portanto, importante explorar as oportunidades proporcionadas pelas novas tecnologias para desenvolver soluções inovadoras que ajudem a atingir este objetivo. Assim nasce o projeto da carteira digital como solução financeira que tem como objetivo a total, ou quase total, desmaterialização do dinheiro em países onde atualmente a maioria das transações financeiras envolvem pagamentos em dinheiro vivo.

Este documento é o relatório final do projeto Carteira Digital como Solução Financeira, um projeto de engenharia informática realizado na empresa Innovation Makers durante 9 meses, no contexto do Mestrado em Engenharia Informática.

Apresenta-se o contexto social em que o projeto se integra, dando especial ênfase ao problema que este endereça e às oportunidades associadas, e também o contexto técnico, nomeadamente o sistema de pagamentos existente, para utilizadores com contas bancárias. É fornecida uma visão global de alto nível do trabalho realizado, o qual tem duas partes distintas. A primeira parte do trabalho centra-se na conceção e na concretização de uma solução para estender o sistema de pagamentos existente de forma a alargar as movimentações financeiras aos utilizadores que, não possuindo contas bancárias, são donos de pelo menos uma carteira digital. Esta solução passa pelo desenvolvimento de um novo componente—o núcleo bancário—responsável pela movimentação de entradas e saídas de dinheiro, tal como movimentações entre carteiras do próprio sistema. A segunda parte do trabalho foca-se na interoperabilidade entre os sistemas de pagamentos. Pretende-se, recorrendo à utilização da tecnologia de *blockchain*, ter uma solução em que as movimentações de dinheiro não dependem de uma entidade central, mas antes do consenso entre as entidades envolvidas. É apresentada a arquitetura da solução concebida para o núcleo bancário e para o sistema de pagamentos e são analisadas e discutidas as decisões mais importantes. Apresenta-se, por fim, o trabalho futuro.

**Palavras-chave:** inclusão social, inclusão financeira, sistema financeiro, carteira digital, interoperabilidade, *blockchain*, economia.

# Abstract

Financial inclusion has become an increasingly debated topic in our society and is recognized as an important goal whose pursuit can boost the economy. It is therefore important to exploit the opportunities offered by new technologies to develop innovative solutions that help achieve this goal. Thus, the Digital Wallet project is born as a financial solution that aims at total or almost total dematerialization of money in countries where currently most financial transactions involve cash payments.

This document is the final report of the Digital Wallet Project as a Financial Solution, an informatics engineering project carried out at the Innovation Makers company for 9 months, in the context of the Master in Informatics Engineering.

It presents the social context in which the project is integrated, giving special emphasis to the problem it addresses and the associated opportunities, as well as the technical context, namely the existing payment system, for users with bank accounts. A high-level overview of the work done, which has two distinct parts, is provided. The first part of this report focuses on the design and implementation of a solution to extend the existing payment system in order to extend financial transactions to users who, having no bank accounts, own at least one digital wallet. This solution involves the development of a new component – the core banking – that is responsible for the movement of inflows and outflows of money, such as movements between the system's own wallets. The second part of this report focuses on interoperability between payment systems. It is intended, using the technology of blockchain, to have a solution in which the money movements do not depend on a central entity, but instead, on the consensus among the entities involved. The architecture of the solution designed for the core banking and the payments system is presented and the most important decisions are analyzed and discussed. This report ends by presenting the future work.

**Keywords:** social inclusion, financial inclusion, financial system, digital wallet, interoperability, blockchain, economy.

# Conteúdo

# Lista de Figuras

# Lista de Tabelas

# Abreviaturas

KYC – Know Your Customers

API - Application Programming Interface

REST – Representational State Transfer

HTTP – Hypertext Transfer Protocol

XML – eXtensible Markup Language

JSON – JavaScript Object Notation

USSD – Unstructured Supplementary Service Data

JPA – Java Persistence API

DLT – Distributed Ledger Technologies

SMS – Short Message Service

NFC – Near Field Communication

QR-Code – Quick Response Code

IBAN – International Bank Account Number

RPC – Remote Procedure Call

SOAP – Simple Object Access Protocol

URI – Uniform Resource Identifier

IB – Internet Banking

PB – Portal Balcão

# Capítulo 1 Introdução

Neste capítulo é apresentada uma introdução ao trabalho desenvolvido. É descrito o contexto do projeto e as motivações para o mesmo. É apresentada a empresa acolhedora, a equipa envolvida e os papeis que estes desempenham no desenrolar do projeto.

## 1.1 Motivação

Atualmente no contexto africano existe a necessidade de aproximar a população dos serviços financeiros de modo a aumentar a literacia financeira e a capacidade de acesso aos serviços pela população em geral. A inclusão financeira tem vindo a ser um tópico em ascensão e é cada vez mais importante junto das grandes organizações neste setor e entre as entidades governamentais. Os níveis de inclusão nestes países rondam os 54%, enquanto que em países da Organização para a Cooperação e o Desenvolvimento Económico (OCDE) estes níveis rondam os 94% (1).

Como se pode ler numa notícia publicada pelo Jornal de Angola, "Mais de metade da população angolana economicamente ativa não possui uma conta bancária e calcula-se que 60 por cento do Produto Interno Bruto (PIB) circula de mão em mão sem passar pelo sistema financeiro do país." (2) E, portanto, "tais recursos, estando fora do sistema financeiro, não geram impostos" (2), o que limita o crescimento económico do país como um todo. É, portanto, importante facilitar o acesso aos serviços financeiros existentes e criar novos sistemas inovadores onde o acesso seja simples.

Com a grande disseminação das tecnologias móveis, em particular dos telemóveis de segunda geração e *smartphones* e também das redes móveis associadas a estes dispositivos, proporcionou-se um meio com elevado potencial para o desenvolvimento de soluções inovadores no sector financeiro. É assim que surge o *Digital Wallet*, um sistema que tem como objetivo suportar a total, ou quase, total desmaterialização do dinheiro em países como Angola, onde a maioria da população tem dificuldade em aceder aos serviços financeiros tradicionalmente prestados pelos bancos. Mais precisamente, o objetivo deste projeto de mestrado é desenvolver uma plataforma eletrónica que permite que a população que não possui contas no banco também realize movimentações de dinheiro de forma segura e fácil. O acesso a estes serviços é assente

no conceito de carteiras digitais (3), as quais funcionam como contas bancárias (do tipo deposito à ordem) mas que se encontram associadas ao número de telemóvel. É através destas carteiras que é realizada a identificação do utilizador e a autorização das movimentações solicitadas.

Desta forma o *Digital Wallet* possibilitará à população ter acesso aos serviços financeiros e promove-se assim inclusão social e financeira dos utilizadores com menos literacia financeira.

## 1.2 Contexto

O sistema *Digital Wallet* foi proposto e desenvolvido na *Innovation Makers* (4), uma empresa portuguesa especializada no desenvolvimento e implementação de soluções inovadoras de *software* e *hardware* centradas na experiência do utilizador.

Toda a atividade da empresa é centrada na criação de soluções tecnologicamente evoluídas para responder a necessidades especificas dos clientes recorrendo a fortes fontes de trabalho qualificado na área da engenharia informática, programação de *software* e *design* de interfaces gráficas.

Entre as múltiplas atividades desempenhadas pela empresa acolhedora, uma destas atividades passa pelo desenvolvimento de soluções para instituições financeiras presentes em países que se encontram em desenvolvimento, nomeadamente para a banca angolana. O sistema *Digital Wallet* encontra-se inserido no contexto de uma destas soluções e pretende-se alargar as funcionalidades da solução existente ao mesmo tempo tirar partido dos componentes que compõem esta solução.

O sistema *Digital Wallet* tem como objetivo fornecer aos clientes um novo método de movimentação de dinheiro cujo destino ou origem são carteiras digitais. Estas carteiras estão associadas ao número de telemóvel dos utilizadores. A solução envolve desenvolver um novo núcleo bancário que aprovisiona estas carteiras digitais ou contas *wallet*. Este núcleo é responsável pela gestão das transações financeiras sobre as contas *wallet* e da gestão dos utilizadores, e, portanto, necessita de armazenar informação sobre as movimentações de dinheiro entre contas internas – *wallets* – e contras externas.

Pretende-se então, com o desenvolvimento deste núcleo, aumentar as funcionalidades do sistema financeiro da entidade acolhedora e consequentemente melhorar a qualidade dos produtos dos seus clientes.

## 1.3 Objetivos

De uma forma genérica, o objetivo deste projeto é contribuir para o desenvolvimento de sistemas de *software* que venham a contribuir para a modernização

i) da forma de movimentação do dinheiro nos países em desenvolvimento onde atualmente a principal forma de movimentar dinheiro é de mão em mão e ii) investigação sobre a criação de um sistema de pagamentos que fornece a interoperabilidade entre entidades de pagamento sem necessitar de um agente intermediário ou entidade central.

Em concreto, existem dois grandes objetivos a atingir na realização deste projeto:

- A conceção e desenvolvimento do componente *Núcleo-Bancário* que vai permitir suportar movimentações de dinheiro entre carteiras digitais através do telemóvel. Os utilizadores devem ser capazes de realizar pagamentos e transferências de ativos de e para carteiras digitais. Desta forma é possível diminuir a barreira de aprendizagem sobre os serviços do sector bancário e aumentar a literacia financeira dos utilizadores.
- A conceção e desenvolvimento de uma plataforma distribuída e descentralizada que permite a comutação, aceitação e reconciliação financeira. Este sistema visa a remoção de agentes intermediários na reconciliação financeira entre as entidades de pagamento. Desta forma é possível diminuir tanto as taxas cobradas nas transferências interbancárias, quanto a duração da transferência.

## 1.4 Contribuições

O sistema *Digital Wallet* pretende contribuir para uma redução da desigualdade entre as várias camadas do sistema financeiro presente, principalmente nos países em desenvolvimento onde esta discrepância é cada vez mais acentuada (5). Este sistema visa implantar a filosofia de carteiras digitais nestes países. É com esta visão que se utilizam as carteiras digitais como veículo que permite o acesso aos sistemas financeiros de forma mais simplificada e de fácil utilização. Para combater esta desigualdade, pretende-se com o sistema *Digital Wallet* reduzir as barreiras de entrada dos cidadãos com menos literacia financeira e a população que não possui contas nas instituições, aos serviços financeiros.

O desenvolvimento do sistema *Digital Wallet* contribuiu para enriquecimento das funcionalidades dos sistemas que já existiam a priori deste projeto de tese. Oferecendo para os clientes da instituição financeira, no qual este projeto encontra-se contextualizado, uma nova forma de transacionar os seus ativos financeiros.

No que diz respeito a construção do sistema de pagamentos interoperável, embora em contexto de investigação, foi possível verificar que na prática é possível existir um sistema que seja descentralizado e distribuído, onde as transações entre as entidades da rede sejam realizadas e finalizadas em tempo real.

## 1.5 Organização do documento

Este documento está organizado da seguinte forma:

- Capítulo 2 – Apresenta o sistema Digital Wallet, o domínio do problema do sistema e discute alguns trabalhos relacionados. São também expostos conceitos importantes do negócio e apresentados os casos de uso do sistema.
- Capítulo 3 – Apresenta o processo de conceção e construção do componente *Núcleo-bancário Digital Wallet*. É apresentada a metodologia utilizada durante a conceptualização e desenvolvimento deste. São apresentadas múltiplas vistas da arquitetura do sistema e componente para uma melhor compreensão. Por fim são apresentados os múltiplos ambientes e os testes feitos sobre o núcleo.
- Capítulo 4 – Apresenta o sistema de pagamentos interoperável. É analisado o problema da interoperabilidade presente nos atuais sistemas de pagamentos e proposta uma possível solução que recorre a utilização de tecnologias de *ledgers* distribuídos (6), mais especificamente a *ledgers* do tipo *blockchain* (7).
- Capítulo 5 – Discute as principais contribuições e as mais valias dos sistemas desenvolvidos. São apresentadas as competências adquiridas durante a execução dos mesmos. Por fim, discute-se trabalho futuro e possíveis melhorias.

# Capítulo 2 Sistema *Digital Wallet*

Neste capítulo é descrito de forma detalhada o principal problema que este projeto de tese endereça e a solução estudada e desenhada para resolver o mesmo. O capítulo começa por apresentar o sistema financeiro existente da empresa acolhedora. Depois, fornece uma visão geral da solução, nomeadamente do conceito principal em que a solução assenta, e é analisado como a solução impacta este sistema. O capítulo termina com a análise das soluções existentes que estão mais relacionadas com este projeto.

## 2.1 Contexto

Neste documento, o nome *Digital Wallet* designa o sistema de *software* que alarga as movimentações financeiras aos utilizadores que, não possuindo contas bancárias, são donos de pelo menos uma carteira digital (3). Este sistema é uma versão estendida de um sistema já existente centrado apenas em utilizadores com conta bancária e que permite aos utilizadores acederem aos serviços bancários através de aplicações web e mobile.

O Sistema *Digital Wallet* encontra-se enquadrado no produto que é vendido e suportado para múltiplas instituições financeiras, por entendendo-se por instituições financeiras empresas cuja atividade consiste em realizar investimentos, empréstimos e financiamento e onde esta instituição é supervisionada pelo Banco Central (8) do país. Assim pode-se entender por instituição financeira todos os bancos que pertençam às seguintes categorias: banco de investimento, banco comercial e banco de desenvolvimento. O sistema *Digital Wallet* encontra-se contextualizado em um banco de investimentos angolano.

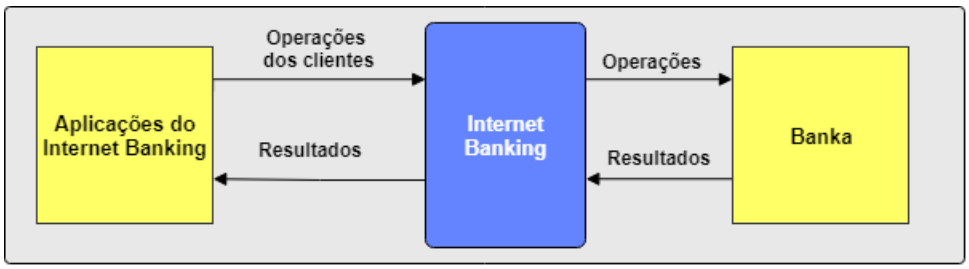


*Figura 1 - Diagrama de contexto do componente Internet Banking anterior ao sistema Digital Wallet*

A**Erro! A origem da referência não foi encontrada.** Figura 1 apresenta o d iagrama de contexto do sistema Internet Banking interno da entidade acolhedora anterior ao sistema *Digital Wallet.* Neste caso é possível ver que a principal fonte de dados é o componente Banka, que é o único aprovisionador de contas bancárias e consequentemente de dinheiro e das transações financeiras. Pode-se ver na Figura 2 que

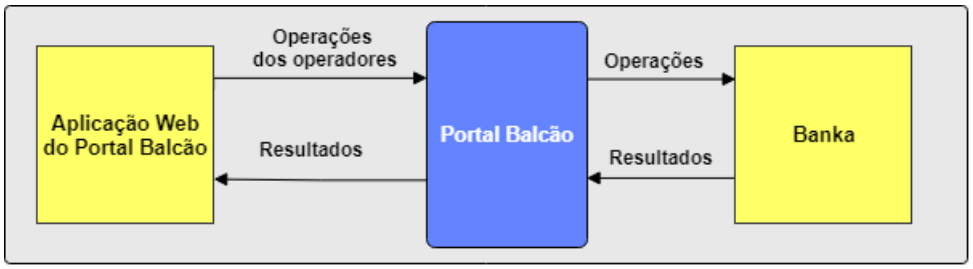


*Figura 2 - Diagrama de contexto do componente Portal Balcão anterior ao sistema Digital Wallet*

o mesmo acontece com o componente Portal Balcão.

### 2.1.1 Catálogo de elementos

***Banka***

A Banka é o elemento responsável por aprovisionar as contas da instituição financeira e fornece serviços básicos sobre as contas bancárias, como consulta de saldo e transferência entre contas. A Banka é considerada um ponto único de conhecimento das contas, sendo este ponto utilizado por todos os elementos que necessitam de obter conhecimento sobre essas contas e também sobre os utilizadores e donos das mesmas.

Diferente dos outros elementos, a Banka existe no interior da instituição financeira e não é desenvolvido pela empresa acolhedora. Este elemento apenas fornece uma interface para que os outros componentes do sistema possam aceder as suas funcionalidades.

***Aplicações do Internet Banking***

Os utilizadores da instituição bancária acedem aos serviços financeiros através da utilização das aplicações. Estas aplicações podem ser web através do sítio online da área do cliente que a instituição financeira disponibiliza ou através das aplicações mobile disponíveis na *Play Store* (9) e *App Store* (10) para os sistemas operativos Android (11) e iOS (12) respetivamente.

***Internet Banking***

As aplicações fazem pedidos ao Internet *Banking* que é desenvolvido na instituição acolhedora e que fornece uma gama de serviços financeiros para que os utilizadores

possam fazer a gestão das suas contas, visualização dos ativos e passivos bem como pagamentos e transferências. Todas as operações que envolvam a movimentação de dinheiro implicam um pedido aos serviços da Banka.

***Aplicação Portal Balcão***

As instituições financeiras, normalmente, possuem ramos com balcões onde os utilizadores se podem dirigir para gerirem as suas informações e as suas contas. Denomina-se então por aplicação Portal Balcão a aplicação web que é executada no computador associado ao balcão. Os utilizadores dos balcões são denominados por operadores de balcão.

***Portal Balcão***

Os operadores podem realizar consultas e alterações sobre os dados dos utilizadores e criar novas adesões. Estes pedidos são feitos ao Portal Balcão que é o *middleware* que fornece serviços aos vários balcões existentes.

## 2.2 Visão geral da solução

Como mencionado anteriormente, a solução é desenvolvida em torno do conceito de *carteira digital*. Estas carteiras funcionam como contas bancárias do tipo depósito à ordem, mas que não se encontram associadas a nenhuma instituição financeira. Este facto implica a não necessidade do cliente se locomover ao balcão para fornecer os seus dados pessoais para criar a sua conta. De forma a que clientes que vivem em zonas onde não existem balcões e/ou não exista acesso a internet (como zonas rurais e zonas de difícil acesso) possam obter contas, efetuar pagamentos e transferências de maneira fácil e rápida, a solução desenvolvida tira partido da unicidade do número de telefone do cliente e considera que este número identifica o cliente como dono único de uma carteira digital.

A solução passa por estender o sistema existente de forma a disponibilizar um conjunto adicional de funcionalidades.

Estas novas funcionalidades têm como objetivo:

- Aumentar o leque de serviços financeiros fornecidos pela instituição financeira.
- Facilitar a atração de novos clientes pela instituição financeira onde o sistema *Digital Wallet* irá ser integrado, oferecendo um serviço que é de fácil utilização e que abrange uma maior gama de tecnologias.
- Disponibilizar um novo tipo de conta que os utilizadores que já são clientes da instituição financeira podem tirar partido para as suas transações financeiras.

Para a concretização da extensão pretendida, foi proposta a criação de um novo componente, designado por *Núcleo-Bancário*. Este novo componente irá aprovisionar informação sobre este novo tipo de contas —as contas *wallet*— e ser responsável por todas as transações, gestão e históricos destas novas contas. O componente expõe os seus serviços através de uma interface que permite que os outros componentes do sistema *Digital Wallet* possam tirar partido destas novas funcionalidades.

O diagrama de contexto para este novo componente do sistema *Digital Wallet* é apresentado na Figura 3.

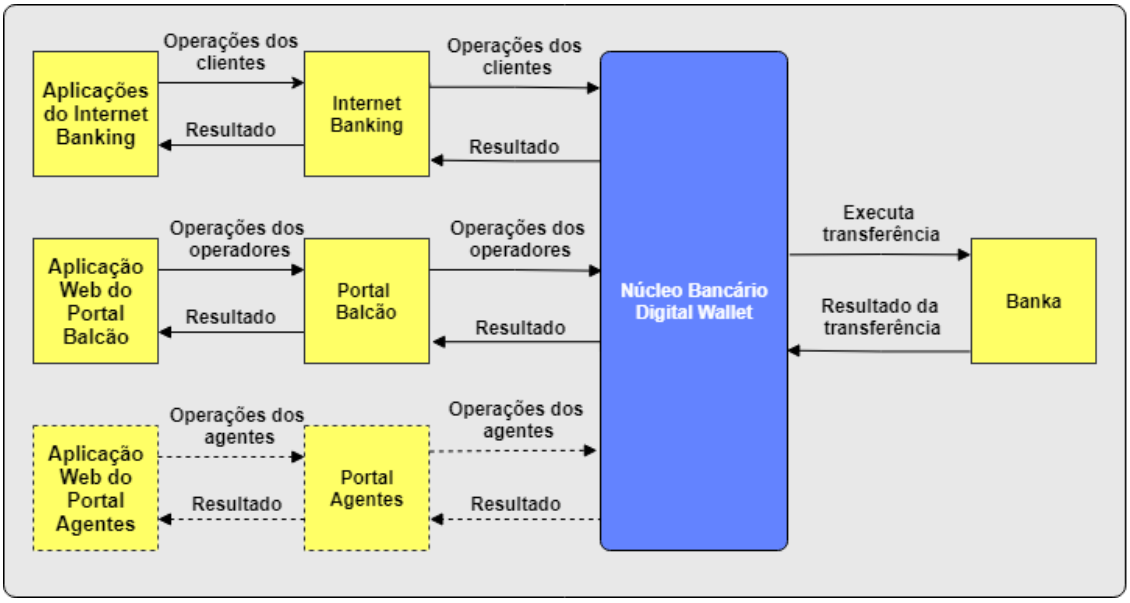


*Figura 3 - Diagrama de contexto do sistema Digital Wallet*

### 2.2.1 Alterações necessárias

Com a introdução de um novo componente é necessário alterar os outros componentes existentes para que estes possam dar suporte para os utilizadores.

***Internet Banking***

De forma a garantir uma otimização da forma como são disponibilizadas as novas funcionalidades, é preciso fornecer estas através dos seguintes canais: *Unstructured Supplementary Service Data* (USSD) (13) que permite consultas e movimentações; *Short Message Service* (SMS) que permite consultas; aplicações móveis Android e iOS e aplicações Web que fornecem todos os serviços disponíveis. Desta forma enumera-se as seguintes alterações a serem feitas neste componente:

- Tendo em consideração que atualmente só são feitas transações nos canais servidos através de *HyperText Transfer Protocol Secure* (HTTPS) (14), não pode ser

utilizado o mecanismo de envio de credenciais de segundo nível (segundo passo que acrescenta uma camada de segurança ao processo de autenticação do utilizador. Serve para garantir que este é iniciador da transação.) através do SMS Token - que o utilizador recebe através de uma SMS, para validar uma movimentação em curso - passa a ser necessário a utilização de credenciais distribuídas previamente aos clientes. Esta credencial apenas é necessária para poder validar a movimentação caso esta esteja a ser feita através do canal USSD.

- As credenciais de acesso de primeiro nível dos canais USSD e SMS são garantidas através da unicidade do número de telefone do cliente que se encontra a realizar a operação e dessa forma é possível fazer a autenticação de primeiro nível de forma totalmente automática e transparente para o utilizador.
- O acesso de primeiro nível dos canais Android, iOS e Web continua a ser feito através do tradicional *username* e *password*. No entanto torna-se necessário haver uma zona de associação para clientes que apenas possuam contas *wallet.*
- Para utilizadores que são clientes da instituição financeira e que estejam na área autenticada das aplicações, deve ser possível criar contas *wallet* e associar estas ao perfil do utilizador.
- Para utilizadores que não são clientes da instituição financeira e que desejem possuir contas do tipo *wallet*, deve existir na área não autenticada uma zona que permite ao utilizador criar uma nova conta *wallet* e associar um *username* e *password* para que este possa usufruir das aplicações.
- Os utilizadores que estejam na área autenticada dos canais Android, iOS ou Web, devem poder ter acesso às contas *wallet* em todos os locais onde é possível aceder as contas que a instituição financeira possui. Nomeadamente deve ser possível aceder as contas wallet nos seguintes menus: menu transferência, menu pagamento, menu consulta de movimentos, menu consulta de saldos, menu consulta de contas e outros que sejam relevantes.
- Deve existir uma nova entrada no menu com suporte para transferências para contas *wallet* onde a origem pode ser ou não outra conta *wallet*. Este menu é importante pois disponibiliza aos utilizadores uma forma de poderem enviar dinheiro para as suas contas *wallet* ou para contas *wallet* de outros utilizadores. Esta transferência deve ser suportada com a introdução do número de telemóvel como o destino da transferência.

***Portal Balcão***

- O utilizador deve ser capaz de depositar numerário diretamente em uma conta *wallet* no balcão da instituição financeira. Esta funcionalidade permite a entrada de dinheiro no sistema *wallet* através do depósito.

- O utilizador deve ser capaz de levantar numerário de uma conta *wallet* no balcão da instituição financeira. Esta funcionalidade permite a saída de dinheiro do sistema *wallet* através do levantamento.
- O balcão deve fornecer serviços de gestão de contas para o utilizador, como o ativar e desativar de uma conta do utilizador caso este tenha mais do que uma conta *wallet.*
- No balcão deve ser possível eliminar contas *wallet* caso o cliente o deseje. Este caso apenas deverá acontecer caso o cliente tenho o saldo todo a zero.
- O utilizador deve ser capaz de aumentar o seu nível de *know your customer* (KYC) (15) no balcão, fornecendo para este efeito as informações necessárias para o nível em causa.
- O utilizador deve ser capaz de solicitar, mediante a apresentação de documentos, o envio de novas credencias para o acesso de segundo nível.

***Portal Agentes***

Este portal ainda não existe, no entanto, em relação às suas funcionalidades, deverá fornecer as mesmas funcionalidades que são disponibilizadas nos balcões, distinguindo-se destes no nível de confiança. O Portal Balcão é operado por funcionários da instituição financeira criando assim uma camada de confiança que não existe no caso do Portal Agentes, que irá ser operado por agentes externos à instituição financeira.

***Banka***

A Banka não deve ser alterada. Todos os mecanismos necessários para a construção do núcleo já são fornecidos pela Banka. Nomeadamente o aprovisionamento de contas e dinheiro e também o fornecimento de funcionalidades relacionadas com a movimentação de dinheiro entre contas.

## 2.3 Trabalho relacionado

O conceito de carteira digital ou *digital wallet* é um conceito que se refere a um dispositivo eletrónico ou serviço online que permite aos seus utilizadores realizarem transações online com o auxílio de um computador ou telemóvel. O objetivo desta filosofia é diminuir o número de passos que o utilizador necessita realizar durante um pagamento e facilitar o processo de compra. Desta forma é necessário tirar partido das múltiplas tecnologias existentes para atingir este objetivo. A tecnologia mais utilizada para realizar pagamentos seguros com o auxílio dos dispositivos moveis é o *Near-field Communication* (NFC) (16) que permite a transmissão de informação apenas com a aproximação entre o telemóvel e o terminal de pagamento.

Com esta definição em mente, é possível idealizar uma miríade de possibilidades e casos de uso possíveis para uma carteira digital, que vai desde o armazenamento de informação pessoal como morada e bilhete de identificação do utilizador, até mesmo o armazenamento de cupões de desconto em lojas.

O conceito de carteira digital surgiu no final dos anos 90s em empresas ligadas aos serviços de pagamentos online durante a época que é conhecida como a explosão *dot-com* (17). No entanto estas carteiras digitais não tiveram o sucesso que apresentam nos dias atuais, não por que não funcionavam, mas porque a necessidade dos consumidores não existia e não havia o reconhecimento tanto a nível de sector financeiro como a nível de publicidade junto do público-alvo.

Recentemente a carteira digital tem sido amplamente explorada em dois grandes domínios:

- Domínio das criptomoedas, de forma a armazenar de forma segura o par de chaves públicas e privadas do utilizador para que este possa aceder às suas múltiplas moedas online.
- Domínio dos cartões bancários, de forma a armazenar os cartões que o utilizador possui em um ponto único de acesso para que este passe a não necessitar de ter o cartão físico para poder realizar transações seja online seja em lojas físicas que aceitem pagamentos via NFC.

A carteira digital no contexto do sistema *Digital Wallet* tira partido da unicidade do número de telefone do cliente que o identifica como dono único de uma carteira digital e permite assim efetuar transações financeiras sobre as mesmas. É necessário salientar que as carteiras existentes no núcleo não estão associadas a nenhum balcão de instituições financeiras, o que possibilita a existência de clientes que podem e conseguem usufruir do sistema e possuir contas sem ter a necessidade de se dirigir a um balcão bancário, sendo assim possível atrair novos clientes que vivem em zonas onde ou não existam balcões ou não exista acesso a internet, como zonas rurais e zonas de difícil acesso.

Apesar de existirem plataformas semelhantes no mercado, o sistema *Digital Wallet* diferencia-se destas plataformas de forma geográfica, já que a grande maioria das soluções existentes não prestam serviços em Angola e os serviços presentes em Angola não oferecem exatamente os mesmos serviços que a solução *Digital Wallet*. A seguir enumeram-se potenciais concorrentes da solução.

A **Wizzit Merchant Platform** (18) pertence à instituição de crédito Wizzit, sediada na África do Sul, e permite aos comerciantes fazerem pagamentos, transferências, levantamento de dinheiro, realizarem depósitos e ainda a venda de vales

pré-pagos. Utiliza ainda a aplicação ZING Mobile Wallet para que os utilizadores possam realizar pagamentos através do seu telemóvel. Estas soluções aparentam ser baseadas em *software* para telemóveis e *tablets*. No entanto, é uma *wallet* sem ligação a nenhuma rede bancária, aceitando pagamentos de e para todo o tipo de instituições bancárias.

A **EzPay Africa** (19) é uma empresa nigeriana que oferece uma lista de serviços baseados em SMS's, IV (20), USSD, INTERNET, NFC e SmartCard (21) para pagamentos de faturas, carregamento de telemóveis, compras ou transações entre pessoas a nível nacional e internacional. Este tipo de carteira não está associado a nenhuma entidade bancária, sendo possível o registo a partir de qualquer conta bancária e o envio e receção de dinheiro a partir de qualquer parte do mundo.

**M-Pesa** (22) é uma *Mobile Wallet* pertencente à Vodafone, e que atua no Quénia, Roménia, Albânia, Tanzânia, Afeganistão, África do Sul e Índia e que permite enviar e receber transferências, realizar carregamentos de serviços de telemóveis e compras em agentes aderentes a este serviço. Funciona como uma carteira digital gerida exclusivamente a partir do telemóvel. Em 2012 cerca de 17 milhões de contas M-Pesa estavam registadas apenas no Quénia, e em 2016 cerca de 7 milhões na Tanzânia. O serviço do M-Pesa, sendo destinado maioritariamente a países em desenvolvimento, baseia-se em códigos USSD para realizar as transações, apesar de suportar também outros sistemas.

O **Meo Wallet** (23) é a Carteira virtual da Meo que permite efetuar pagamentos através do telemóvel, SMS ou lojas online. A aplicação móvel permite efetuar pagamentos através de tecnologias NFC e QR-Code (24). O carregamento de saldo pode ser feito através de transferência bancária, multibanco, ou através de transferência a partir de outra Meo Wallet. Permite também efetuar compras e fazer pagamentos a partir da televisão.

O **Google Wallet** (25) é uma solução da Google (26) que entrou em funcionamento em janeiro de 2015 e pode ser usado em várias cadeias de retalho ou superfícies aderentes ao serviço. Estas carteiras podem ser usadas em qualquer loja aderente ao serviço Mastercard PayPass (27) (que são terminais de pagamento localizados em lojas, normalmente utilizados para passar cartões de crédito que suportem o Google Wallet). No entanto nem todos os terminais PayPass permitem a utilização do Google Wallet. É uma aplicação criada para possibilitar a transferência de dinheiro entre dispositivos Android e iOS.

O **MB Way** (28) é a aplicação oficial da SIBS (29) que permite usar os serviços MB Way e MB Net. Trata-se de uma solução interbancária para fazer transferências imediatas e compras. Permite realizar compras através da utilização de um cartão virtual

MB Net ou cartões bancários virtuais associados à conta bancária, em comerciantes que aceitem pagamentos com American Express, MasterCard e Visa. Também permite pagar apenas com o número de telemóvel nos comerciantes aderentes ao MB Way e efetuar levantamentos de dinheiro sem cartão, através da ordem de levantamento em um terminal ATM (30) a partir do telemóvel.

## 2.4 Conclusão

Este capítulo apresenta os múltiplos componentes que integram o sistema *Digital Wallet* e como estes estão relacionados.

A solução passa então pela criação de um novo núcleo-bancário que irá aprovisionar as novas contas *wallets* e é responsável pelas transações sobre estas. Devido a introdução do núcleo e nas novas carteiras é proposto uma série de alterações que devem ser efetuadas sobre os restantes componentes de forma a suportarem as novas contas.

O capítulo termina então por apresentar algumas soluções relacionadas que tiram partido do conceito de carteiras digitais. No entanto estas soluções oferecem serviços distintos dos que são oferecidos pelo sistema *Digital Wallet* ou não estão presentes no país para qual este projeto se destina.

# Capítulo 3 Núcleo-bancário *Digital Wallet*

Neste capítulo é descrito de forma detalhada o *núcleo-bancário* do sistema *Digital Wallet*. O capítulo começa por apresentar a metodologia de trabalho utilizada e a sua implementação durante a da criação desta solução. De seguida apresenta os casos de uso definidos para os principais intervenientes do sistema. Apresenta uma visão geral do sistema, descreve as principais decisões, nomeadamente as associadas às tecnologias a usar. Apresenta uma breve descrição sobre os serviços que são fornecidos pelo núcleo para responder aos casos de uso definidos para o sistema e uma solução que facilita a integração do núcleo em outros componentes. Enumera-se vistas arquiteturais sobre o núcleo-bancário de modo a facilitar a compreensão do sistema e termina-se com a apresentação dos ambientes e testes realizados sobre o mesmo.

## 3.1 Metodologia de trabalho

Nesta secção pretende-se apresentar a metodologia de trabalho utilizada para o desenvolvimento e correções do *núcleo-bancário.* O sistema *Digital Wallet*, tal como referido no Capítulo 2, é a composição de múltiplos subcomponentes que em conjunto realizam as funções e disponibilizam serviços na área financeira. O *núcleo-bancário* devido à sua natureza independente e devido ao facto de ser um componente desenvolvido durante um projeto de mestrado, encontra-se à parte dos outros em termos de metodologias de trabalho.

Atualmente existe uma miríade de métodos de trabalho na área de desenvolvimento de produto, contudo todos seguem uma mesma estrutura básica:

- Análise do problema e definição dos objetivos a serem alcançados.
- Traçar um plano para solucionar o problema e alcançar os objetivos definidos.
- Executar o plano, avaliar e verificar se a solução resolve o problema e se os objetivos foram alcançados.

Para o desenvolvimento do *núcleo-bancário* foi utilizado o modelo incremental (Figura 4) (31). Este modelo caracteriza-se por múltiplos ciclos de incrementos realizados sobre o produto em desenvolvimento.

O primeiro ciclo é o mais importante para o sucesso do produto, e caracteriza-se por fornecer ao cliente, num curto espaço de tempo, um produto piloto com o conjunto de funcionalidades centrais e requisitos básicos implementados.

Os ciclos seguintes são caracterizados pelo acréscimo de novas funcionalidades. São realizados tantos ciclos quantos os necessários até o produto atingir a sua versão final.

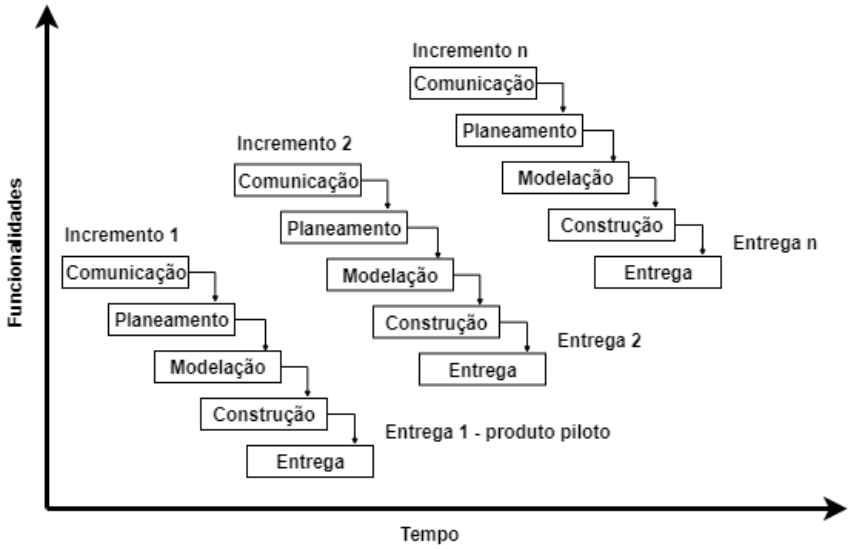


*Figura 4 – Modelo incremental*

Devido à complexidade associada ao desenvolvimento de *software* os modelos dividem o processo em etapas bem estruturadas com tarefas mais pequenas para serem geridas mais facilmente. Estas etapas estabelecem momentos onde devem ser tomadas decisões importantes durante o ciclo de vida do *software*.

O modelo incremental não é uma exceção. Cada ciclo é constituído por cinco etapas:

- Comunicação: Inicia o ciclo com a descrição do problema para qual o *software* deve solucionar e descreve as necessidades dos utilizadores e clientes.
- Planeamento: Descreve as tarefas que pretendem solucionar o problema definido na etapa anterior e estipula o calendário para este ciclo com datas de início e fim; atrasos associados.
- Modelação: Desenho da arquitetura de software e estrutura da solução, estabelecendo assim uma relação entre a solução e o problema. Define as entidades envolvidas tais como estrutura de dados, camadas de *software*, interfaces e algoritmos.
- Construção: Constitui na implementação do modelo definido na etapa anterior e os respetivos testes associados para a validação e verificação das funcionalidades em função dos requisitos levantados na etapa de comunicação.

- Entrega: Consiste na instalação do *software* no cliente para que este possa ser validado pelo mesmo e inclui o apoio e a manutenção do produto.

O *núcleo-bancário* foi desenvolvido em dois incrementos com a duração de cinco meses e um mês respetivamente. O primeiro incremento corresponde ao desenvolvimento das funcionalidades principais do núcleo com a totalidade das funcionalidades principais implementadas (com exceção das funcionalidades relacionadas com os agentes, que não estão incluídas neste projeto de mestrado). O segundo incremento corresponde à etapa de correções e implementações de funcionalidades que eram necessárias na integração dos componentes e que não foram contempladas no incremento anterior.

- A comunicação em ambas as etapas foram feitas principalmente com a marcação de reuniões e *e-mail* junto dos envolvidos no sistema Digital Wallet, nomeadamente, equipas de desenvolvedores dos componentes Internet Banking e Portal Balcão, supervisor do projeto de mestrado, gestores e engenheiro chefe da área de *middleware* (32).
- Planeamento feito normalmente com o supervisor e gestores através de marcações de reuniões periódicas durante esta etapa.
- Modelação executada através de reuniões com o supervisor e engenheiro chefe da área de *middleware.*
- A construção do componente foi realizada principalmente de forma autónoma, com pequenas reuniões intercalares com o supervisor para dúvidas sobre o domínio do problema ou com o engenheiro chefe para dúvidas sobre as tecnologias e integração da Banka no núcleo.
- As entregas consistem na instalação do núcleo nos ambientes de testes para que tanto as outras equipas possam integrar com os seus componentes os serviços do núcleo como para a equipa de testes poderem validar junto aos requisitos levantados na etapa de comunicação se o núcleo implementa as especificações.

Para organizar e manter a atualização do estado, gastos envolvidos – principalmente a nível de recursos humanos – e celeridade do projeto, foi utilizado uma ferramenta de gestão de projetos e de tarefas, o Redmine (33). Esta ferramenta permite criar tarefas atribuídas aos colaboradores e a atualização do estado destas para extrair dados de gestão, como o tempo que cada tarefa demorou e consequentemente quais as tarefas mais custosas.

## 3.2 Domínio

Nesta secção pretende-se apresentar os principais conceitos associados ao domínio do sistema *Digital Wallet* e consequentemente do *núcleo-bancário*. Estes conceitos são

apresentados na forma de um modelo de domínio que mostra de forma visual como se relacionam estes conceitos (Figura 5) e de seguida uma breve descrição de cada um dos conceitos.

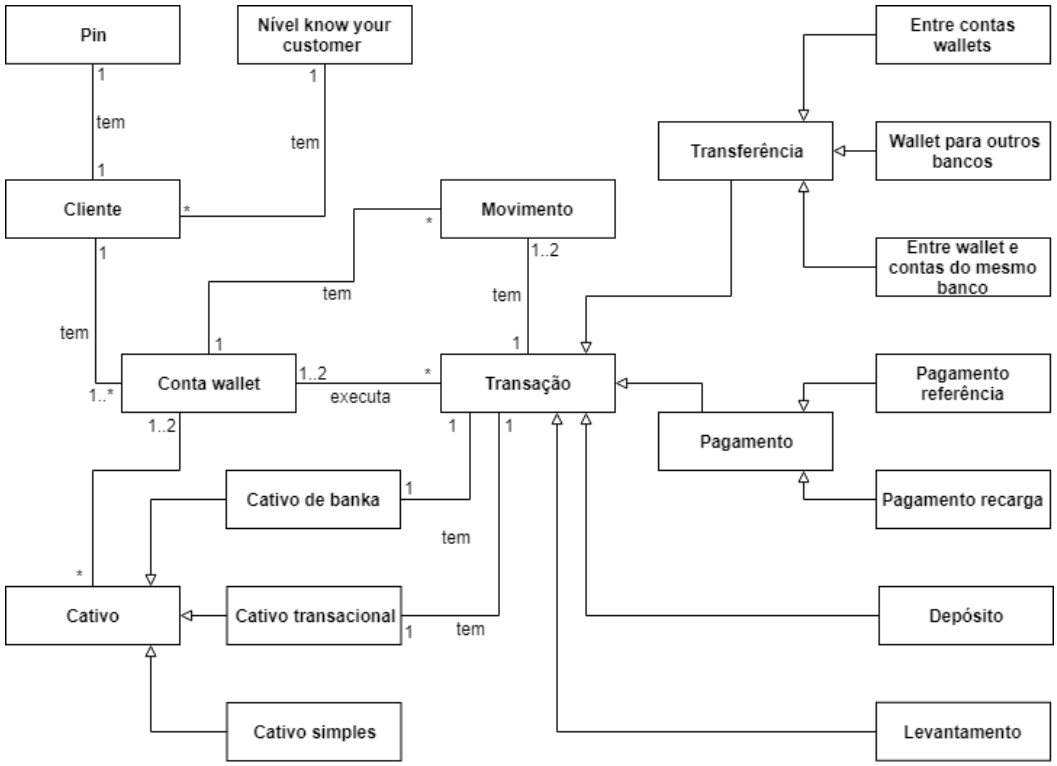


*Figura 5 - Diagrama modelo de domínio do sistema Digital Wallet*

### 3.2.1 Conceitos

#### *Cliente*

O cliente representa um utilizador final do sistema, que possui pelo menos uma conta *wallet* e encontra-se associado a um pin e um nível *know your customer*.

#### *Pin*

O pin é uma sequência numérica de tamanho fixo, que permite ao cliente realizar transações sobre as suas contas *wallet* em canais – canal USSD – onde não é possível realizar a autenticação de segundo nível através do envio de uma mensagem de texto com um *token* para o telemóvel do cliente.

#### *Nível know your customer*

O nível *know your customer* impõe limites às transações do utilizador. Estes limites encontram-se nas seguintes formas:

- Limite no número máximo de transações que o cliente pode efetuar num mês.
- Limite no montante máximo que o cliente pode efetuar numa transação.
- Limite no montante acumulado máximo que o cliente pode efetuar num mês.
- Limite para saldo máximo que o cliente pode ter em suas contas *wallet*.

Estes níveis aumentam de acordo com a quantidade de informação que o sistema possui sobre o cliente. Desta forma o cliente será influenciado a fornecer mais dados para poder ter acesso a limites maiores e irá aumentar a confiança entre as partes.

***Conta wallet***

No ato de criação de um novo cliente é criada uma nova conta *wallet* e associada ao cliente. Esta conta permite ao cliente armazenar dinheiro e efetuar transações com o montante armazenado. Estas contas apenas existem dentro do sistema Digital Wallet.

***Movimento***

O movimento representa uma saída ou entrada de dinheiro de uma *conta wallet*. Este é criado e associado no momento em quem uma transação é sucedida. O movimento armazena dados como a data da realização, montante transacionado, o tipo de transação e se foi saída ou entrada de dinheiro.

***Transação***

Uma transação representa uma movimentação de dinheiro. É um conceito abstrato do sistema que pode representar uma transferência, um pagamento, um cativo, um depósito ou um levantamento. Associado à transação existe um montante que é transacionado e uma conta que efetua a movimentação.

***Transferência***

Uma transferência representa a movimentação entre contas. Estas movimentações podem ser de três tipos:

- Entre contas *wallet*, onde a origem e o destino são contas *wallets.*
- Entre contas internas ao banco, neste tipo de transferência existe uma conta *wallet* e uma conta interna da instituição financeira, mas que não é do tipo *wallet*. Este tipo de transferências permitem a entrada e saída de dinheiro do *núcleo-bancário.*
- Transferências interbancárias onde a origem é uma conta *wallet* e destino é uma conta que pertence a uma outra instituição financeira.

***Pagamento***

O sistema Digital Wallet permite que os clientes possam efetuar pagamentos com as suas contas *wallets*. Os pagamentos podem ser:

- Pagamento entidade e referência, onde a entidade identifica a entidade que irá receber o montante e a referência identifica a fatura que se encontra sobre pagamento.
- Pagamento recarga, o cliente escolhe o tipo de recarga que deseja efetuar e escolhe o valor – normalmente pré-definidos – da recarga.

***Cativo***

Cativo é um montante que se encontra reservado para garantir que uma determinada operação irá ser executada, sendo que este montante fica indisponível para ser utilizado pelo cliente. Podem ser de três tipos:

- Simples, o cativo simples reserva um valor que o cliente indica para não ser utilizado para nenhum objetivo em concreto, na aplicação do cativo, este apenas libera o montante cativado para o saldo autorizado da conta *wallet* que permite ao cliente utilizar para efetuar novas transações.
- Transacional, o cativo transacional reserva um montante associado a uma transação interna ao *núcleo-bancário.* No momento da aplicação do cativo é efetuada a transferência entre a conta *wallet* origem e a conta *wallet* destino.
- Banka, o cativo de banka, reserva um montante associado à uma transação que envolva movimentações na Banka do sistema. No momento da aplicação do cativo este efetua a movimentação na Banka.

***Restrições de integridade***

Nesta subsecção pretende-se informar ao leitor sobre três restrições de integridade do domínio:

- O cativo de banka que se encontra associado a uma transação deve pertencer à mesma conta *wallet* que a transação associada.
- O cativo transacional que se encontra associado a uma transação deve pertencer à mesma conta *wallet* que a transação associada.
- O movimento de uma conta *wallet* tem sempre origem numa transação desta conta *wallet.*

## 3.3 Casos de uso

As operações que o *núcleo-bancário* tem de suportar são obviamente determinados pelas funcionalidades pretendidas para a *Digital Wallet*. Assim sendo, foi necessário começar por fazer um levantamento destas funcionalidades.

Resumidamente, a *Digital Wallet* irá permitir:

- Aos clientes: Realizar operações de movimentação de dinheiro, seja através de transferências ou levantamento e depósitos nos portais balcão
- Aos operadores e agentes: Realizar operações de gestão sobre o cliente, nomeadamente criar novos clientes, aumentar o nível de *know your customer* e operações de depósito e levantamento de numerário.

Nas subsecções seguintes da presente secção apresenta-se detalhadamente os casos de uso que foram definidos de forma a suportar a estas funcionalidades. Começa-se por apresentar os casos de uso associados aos clientes (Figura 6) e de seguida são apresentados os casos de uso associados aos operadores (Figura 7). Não são apresentados casos de uso para os agentes porque os casos de uso associados aos agentes são subconjuntos dos casos dos operadores.

Para simplificar a leitura, entende-se por parceiro tanto os agentes como os operadores.

### 3.3.1 Casos de uso dos clientes


*Figura 6 - Casos de uso do cliente*

***Abrir nova conta wallet***

Cria uma nova conta *wallet* para o cliente. Todas as contas *wallet* estão associadas a clientes que existem no *núcleo-bancário.* Assim existem três situações que pode acontecer durante a criação de uma nova conta *wallet:*

- O cliente já existe no *núcleo-bancário*: Caso o cliente já exista no núcleo a nova conta *wallet* é associada ao cliente e o caso de uso retorna sucesso.
- O cliente não existe no *núcleo-bancário* e é cliente da instituição financeira: Caso ótimo, pois, o cliente é criado no *núcleo-bancário* com o nível máximo de *know your customer* pois se este é cliente da instituição financeira é possível obter todos os dados necessários para aumentar os níveis. Após a criação do cliente é criada uma nova conta *wallet* para o cliente e o caso de uso retorna sucesso.
- O cliente não existe e não é cliente da instituição financeira: Neste caso é criado um novo cliente no *núcleo-bancário* com um nível de *know your customer* de acordo com os dados fornecidos no pedido. Após a criação do cliente é criada uma nova conta *wallet* para o cliente e o caso de uso retorna sucesso.

***Consulta de detalhes das contas wallet***

Operação que permite ao cliente consultar os detalhes da sua conta *wallet*, tal como data de criação, divisa associada a conta, saldo disponível, saldo contabilístico e outros dados que podem ser do interesse do cliente.

***Consulta detalhes do cliente***

Operação que permite ao cliente consultar os detalhes sobre este no sistema, tais como data de criação do cliente, nome, número de telemóvel associado, qual o nível de *know your customer* que este se encontra, documentos associados e outros dados relacionados.

***Consulta histórico de movimentos***

Operação que dado uma conta *wallet* permite ao cliente consultar os movimentos da conta, como data de execução, montante movimentado, saldo da conta após movimentação, se movimento foi crédito ou débito e outros dados. Os movimentos podem ser transferências ou pagamentos.

***Transferências***

Operações que permitem o cliente realizar transferências da sua conta *wallet* para outras contas e vice-versa.

As transferências podem ser de três tipos:

- Entre contas *wallet:* permite ao cliente enviar um determinado montante da sua conta *wallet* para outra conta *wallet* através do número do telemóvel do cliente destino.
- Entre contas *wallet* e contas da instituição financeira: permite ao cliente enviar ou receber um determinado montante entre a sua conta *wallet* para contas que não são *wallets* mas que pertencem a instituição financeira, normalmente são contas do tipo depósito a ordem. Para realizar esta transferência é necessário passar o número da conta destino.
- Conta *wallet* para contas fora da instituição financeira: permite ao cliente enviar um determinado montante da sua conta *wallet* para contas que pertencem a outras instituições financeiras. Neste caso é preciso passar o *International Bank Account Number* (IBAN).

  A única exceção é as transferências entre instituições financeiras e onde o destino é uma conta *wallet*. Este caso não é possível pois o conceito e as contas *wallets* apenas existem dentro do contexto da instituição financeira para qual este projeto se destina.

***Pagamentos***

O sistema suporta dois tipos de pagamentos a partir das suas contas *wallet*. Pagamentos do tipo entidade referência e pagamentos do tipo recarga. Em ambos os casos existem saídas de dinheiro do *núcleo-bancário*, o que implica movimentações de dinheiro na entidade Banka.

***Criar | Executar | Eliminar cativos***

O núcleo permite criar, executar ou eliminar cativos

O núcleo permite criar três tipos de cativos:

- O cativo simples: É um cativo que o utilizador pede para reservar um montante, mas que cujo montante não se encontra associado a nenhuma transação.
- Cativo transacional: É um cativo que esta associado a uma transação entre duas contas *wallet*.
- Cativo de Banka: É um cativo associado a operações que movimenta dinheiro na Banka.

***Consulta histórico de cativos***

Operação que dado uma conta *wallet* permite ao cliente consultar os cativos da conta. Como data de criação e execução, montante cativado, tipo de cativo e outros dados. Os cativos podem ter quatro estados: criado, executado, eliminado ou expirado.

Que representam um ciclo de vida do cativo, sendo os estados eliminados e expirado estados finais do cativo.

***Aumentar nível know your customer***

O cliente através da entidade Internet Banking pode vir a fornecer por livre espontânea vontade dados que o ajudam a aumentar o nível *know your customer*. Quanto mais o núcleo conhecer sobre o cliente, menos restrito é o número de operações mensais que este pode efetuar, tal como os limites de saldo e montantes que este pode movimentar em uma transação.

### 3.3.2 Casos de uso dos parceiros

Nesta secção são apresentadas as funcionalidades que os operadores e agentes podem realizar sobre o *núcleo-bancário*. Tal como foi dito anteriormente, os agentes se distinguem dos operadores por serem funcionários externos à instituição financeira, mas que, no entanto, possuem um nível de confiança que os permitem realizar operações sobre as contas *wallet* e sobre os dados dos utilizadores.

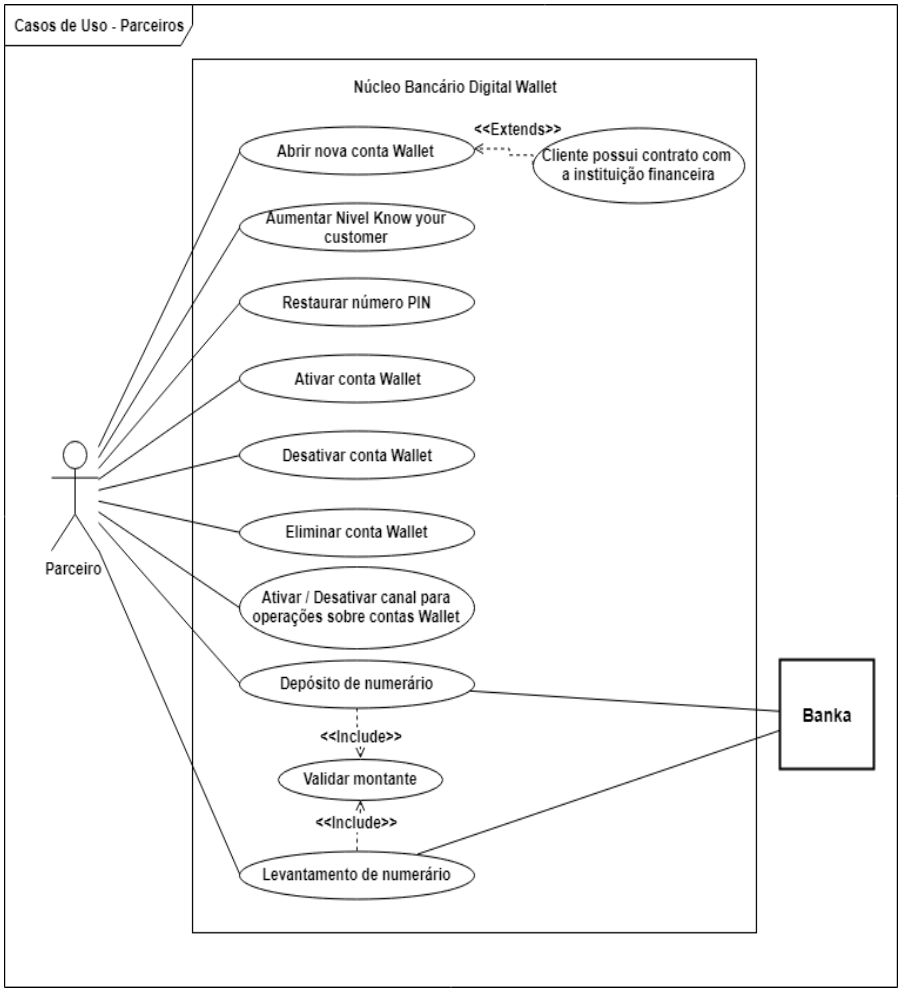


*Figura 7 - Casos de uso dos parceiros*

Tanto os operadores como os agentes estão associados às suas plataformas, Portal Balcão e Portal Agentes respetivamente, estando ambos associados à perfis que são subconjuntos de todas as operações a seguir apresentadas.

É obrigatório que exista um cliente que inicia o caso de uso, pois apenas este é capaz de fornecer o seu número de telemóvel e validar as informações fornecidas.

***Abrir nova conta wallet***

Tal como no caso de uso dos clientes, esta operação serve para criar uma nova conta *wallet* para o cliente. Neste caso a criação é realizada através de um parceiro.

***Restaurar número PIN***

No caso de perda do PIN, o cliente pode se dirigir aos balcões da instituição financeira ou aos agentes e pedir para restaurar o PIN associado ao cliente *wallet*.

***Aumentar nível know your customer***

Tal como no caso de uso dos clientes, esta operação tem como objetivo aumentar o nível de conhecimento sobre o cliente. Neste caso o cliente inicia o processo ao se dirigir a um parceiro.

Um dos requisitos para aumentar para o nível mais elevado é a presença física do cliente junto ao parceiro. Caso o cliente seja cliente da instituição financeira este passo é desnecessário.

***Ativar | Desativar | Eliminar conta wallet***

Operação que permite ao cliente ativar, desativar ou eliminar contas *wallet*. Caso assim o deseje.

***Ativar | Desativar canal para operações sobre contas wallet***

Operação que permite ao cliente ativar, desativar canais onde se pode gerir contas *wallet* e movimentações sobre as mesmas. Um exemplo pode ser desativar o canal USSD para evitar que seja possível realizar movimentações e consultas neste canal.

***Depósito de numerário***

Operação de depósito de numerário em uma conta *wallet*. Um depósito consiste na entrada de dinheiro no *núcleo-bancário.*

***Levantamento de numerário***

Operação de levantamento de numerário de uma conta *wallet.* Um levantamento consiste na saída de dinheiro no *núcleo-bancário.*

## 3.4 Visão geral do sistema

A presente secção pretende apresentar uma visão geral do sistema de forma a facilitar a compreensão e mais concretamente exemplificar a taxonomia do sistema e seus componentes e a forma como estes interagem entre si. Discute-se também as principais escolhas em termos de tecnologia que foram utilizadas para o desenvolvimento do *núcleo-bancário.*

Tal como pode-se ver na Figura 8 o sistema é constituído por cinco componentes principais além do *núcleo-bancário*: o *Internet Banking*, o Portal Balcão, o Portal Agente, a Banka. Cada componente desempenha um papel importante, mais concretamente:

- O Internet Banking é responsável por receber os pedidos dos utilizadores das aplicações *home-banking*.
- O Portal Balcão é responsável por receber os pedidos dos operadores de balcão através da aplicação Portal Balcão.
- O Portal Agentes é responsável por receber os pedidos dos agentes de agências através da aplicação Portal Agentes.
- A Banka é responsável por aprovisionar as contas da instituição financeira e movimentar dinheiro entre estas contas e responsável pela movimentação de dinheiro para outras instituições financeiras.

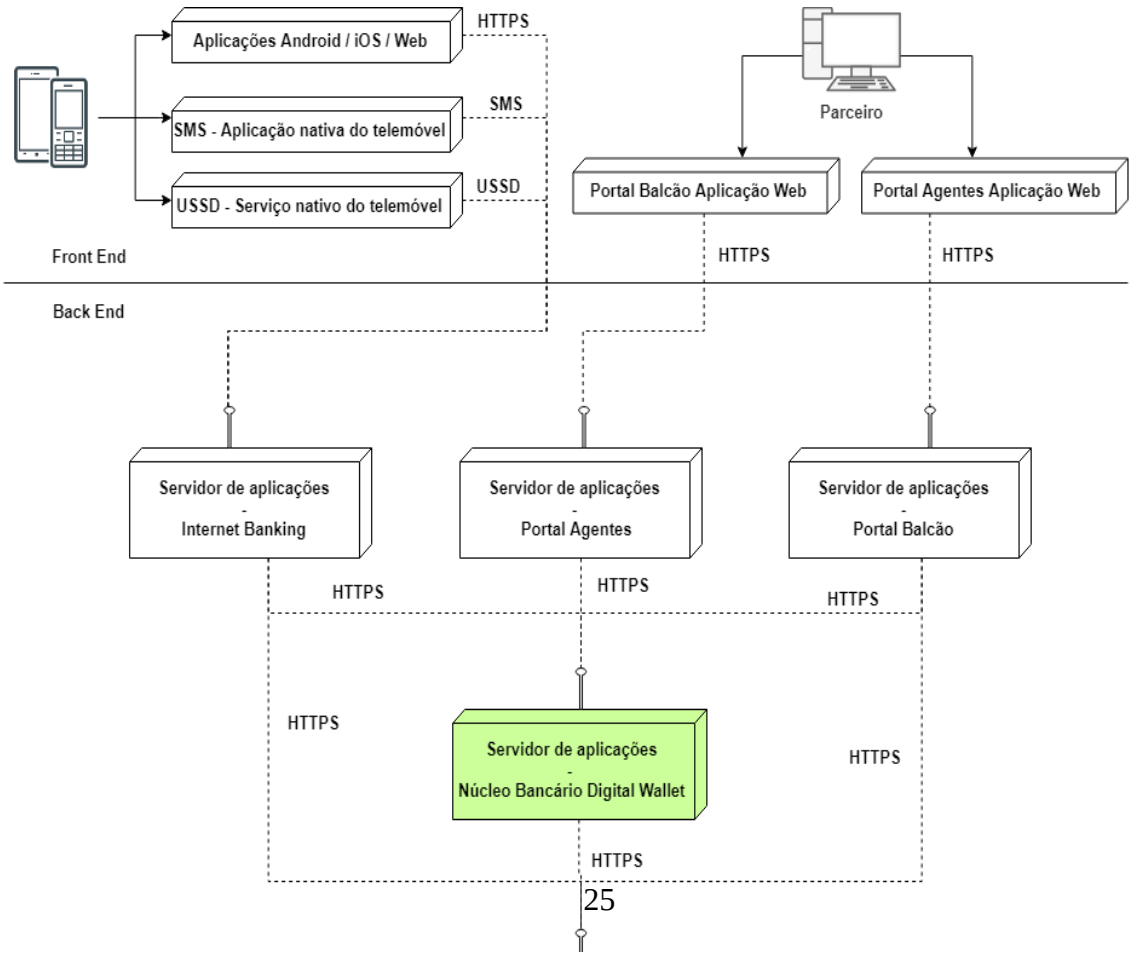


*Figura 8 - Vista de Componentes e Conectores do Sistema Bancário simplificado*

Para desenvolver o *núcleo-bancário* a linguagem de programação escolhida foi o Java (34) com o auxílio do Java EE (35). Esta decisão assenta em dois pontos importantes: a experiência do desenvolvedor adquirida através da educação superior e através da experiência profissional com estas tecnologias; a principal linguagem de programação utilizada na empresa acolhedora é o Java, ou seja, a criação de um novo componente que integra o sistema financeira escrito com a mesma linguagem que os restantes, facilita a alocação de recursos humanos entre projetos. Além disso o Java EE disponibiliza uma série de serviços que simplificam o desenvolvimento de aplicações empresariais, permitindo que os desenvolvedores concentrem a sua atenção nos aspetos funcionais.

O *núcleo-bancário* vai executar dentro de um servidor de aplicações que disponibiliza para o exterior uma interface de serviços à qual os restantes componentes podem fazer pedidos. Esta interface será detalhada na próxima secção.

O servidor de aplicações é necessário para poder disponibilizar para o exterior a aplicação desenvolvida sem que o desenvolvedor precise se preocupar com a complexidade associada à infraestrutura. Este servidor cria uma camada de transparência para o desenvolvedor e permite que este se foque em implementar o sistema que responde às necessidades de negócio da empresa. O servidor de aplicações utilizado para instalar o *núcleo-bancário* é uma versão do WildFly (36) com configurações específicas da empresa acolhedora (cuja descrição está fora do âmbito deste projeto de tese) e que é o servidor utilizado em todos os projetos da empresa acolhedora.

Para persistência dos dados é utilizado uma base de dados relacional da Microsoft SQL Server (37). Este sistema gestor de base de dados é transparente para o desenvolvedor, pois é utilizado o JPA (38) para gerar a camada de acesso aos dados que cria uma camada de abstração sobre a base de dados física. O JPA é uma especificação do Java que descreve um formato normalizado para gerir dados relacionais. Para executar interrogações utiliza-se o *Java Persistence Query Language* (JPQL) (39), uma linguagem para criação de interrogações fortemente influenciada pelo *Structured Query Languange* (SQL) (40), o que facilita a criação de novas interrogações sobre as entidades.

Em relação às operações, estas podem ter duas origens — ou nos clientes, ou nos parceiros. No entanto a execução é exatamente a mesma após a entrada do pedido no *núcleo-bancário*. Por exemplo, uma execução cuja origem seja um cliente, passa pelo seguinte processo: o cliente inicia o pedido, este pedido e dados associados para executar são enviados para o Internet Banking que processa parte do pedido e verifica que deve ser enviado para o *núcleo-bancário*, o pedido é então reencaminhado para o

núcleo que executa a operação. Caso esta operação envolva a Banka, é enviado um pedido para a Banka e a sua resposta persistida em base de dados e só então é enviado o resultado para o Internet Banking, que por sua vez processa o resultado e armazena em base de dados o resultado e só então envia o resultado da operação para o cliente.

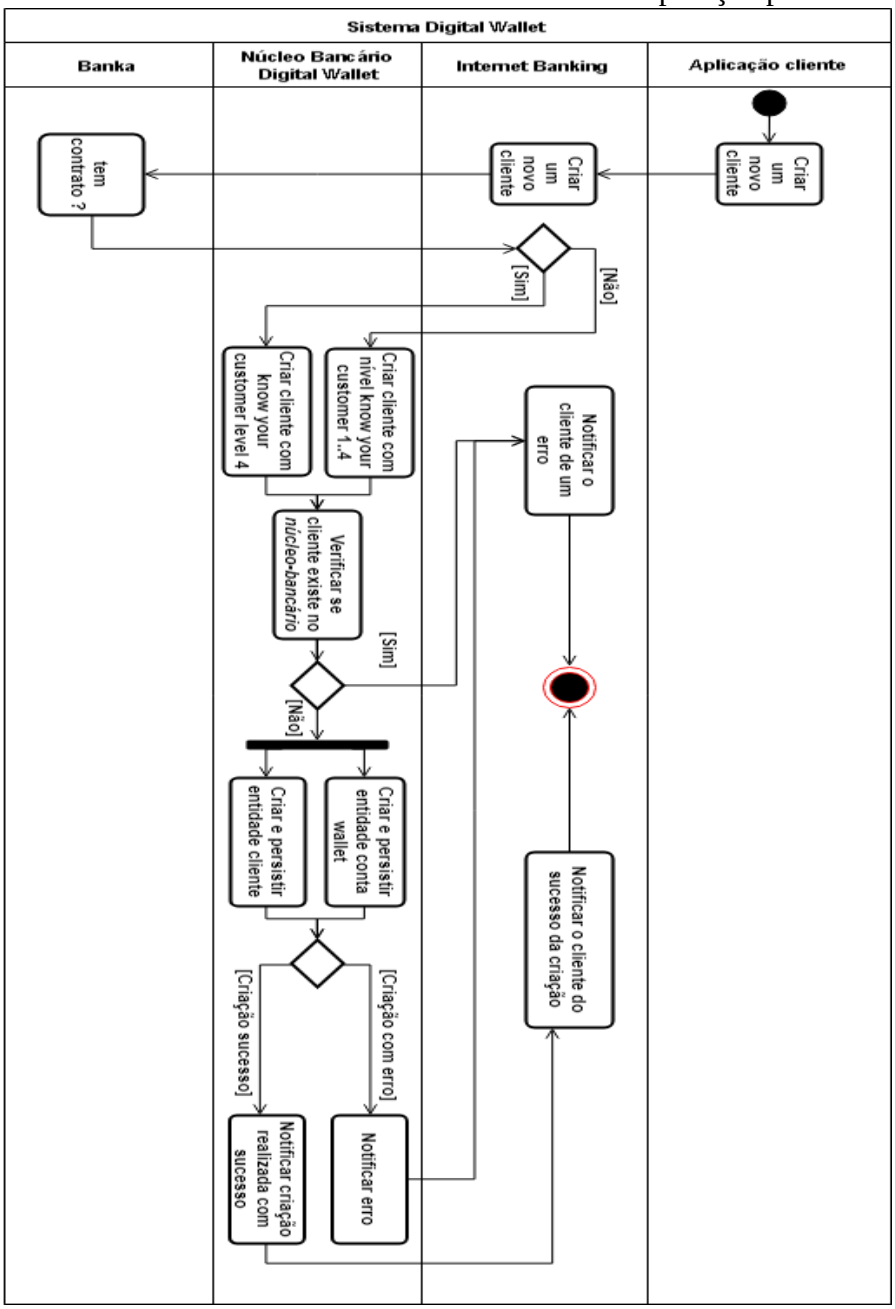


*Figura 9 – Diagrama de atividade do serviço de criar um novo cliente via Internet Banking*

A Figura 9 ilustra a execução de um pedido de criação de um novo cliente onde o cliente inicia o registo através de uma aplicação móvel ou *web*. Por simplicidade, foi excluído a interação com a base de dados e criação de outras entidades.

## 3.5 Serviços

O *núcleo-bancário* irá aprovisionar informação sobre as contas *wallet* e ser responsável por todas as transações, gestão e históricos destas novas contas. Os seus serviços vão ser expostos de forma a poderem ser usados remotamente, pelos outros componentes do sistema *Digital Wallet*, mais precisamente o *Internet Banking*, o Portal Balcão e o Portal Agentes.

Foi decidido expor as funcionalidades do núcleo através de uma interface aplicacional REST (41). Esta decisão assenta sobre alguns pilares:

- Desenho da API: o núcleo-bancário é direcionado aos recursos que este armazena, seja estes recursos clientes, transações, movimentos ou contas. Apenas é possível fazer operações de criar, ler, atualizar e destruir os recursos.
- Escala: O REST não mantém estado sobre o pedido realizado, assim não é necessário manter sessão de clientes do lado do servidor e facilita o escalar do servidor para múltiplos clientes.
- Semântica associada: Os serviços disponibilizados através de uma API REST devem ter uma semântica bem estruturada com auxílio dos verbos HTTP (42). A juncão do tipo de recurso e do verbo faz transparecer uma API que é limpa, visualmente bonita e de fácil entendimento. A simplicidade associada é útil quando se trata de fornecer serviços para outros componentes, como é o caso.
- Facilidade da implementação: O Java EE disponibiliza uma serie de anotações que facilita a implementação desta subcamada em REST desta forma diminui o tempo de implementação e aumenta a produtividade.
- Equipa: A equipa de desenvolvimento presente na empresa acolhedora é familiarizada com o estilo arquitetural REST. O que facilita o transporte de programadores entre projetos e facilita a integração dos serviços do núcleo-bancário.

Concretamente foram implementados dois módulos: O módulo de serviços, que disponibiliza para fora do núcleo uma API REST que pode ser invocada por qualquer cliente — clientes *browser* ou outros componentes — e um módulo conector que implementa os pedidos para a API do lado do cliente. Este modulo tem como objetivo facilitar a integração dos outros componentes, servindo como um conector entre estes.

A decisão por trás do desenvolvimento do módulo conector tem como objetivo, além de facilitar a integração, proteger tanto os componentes que usam as funcionalidades do núcleo como o próprio núcleo para eventuais mudanças na forma como estas funcionalidades são expostas. Soluções parecidas já existiam no sistema para o caso da interação com a Banka, que fornece uma transparência aos componentes que tiram partido das suas funcionalidades.

Em tempo de implementação isto se traduz na importação de um pacote *Java ARchive* (Jar) com o código do conector compilado e pronto para ser utilizado pelo cliente e em tempo de execução traduz-se num pequeno *software* a correr do lado do cliente do núcleo.

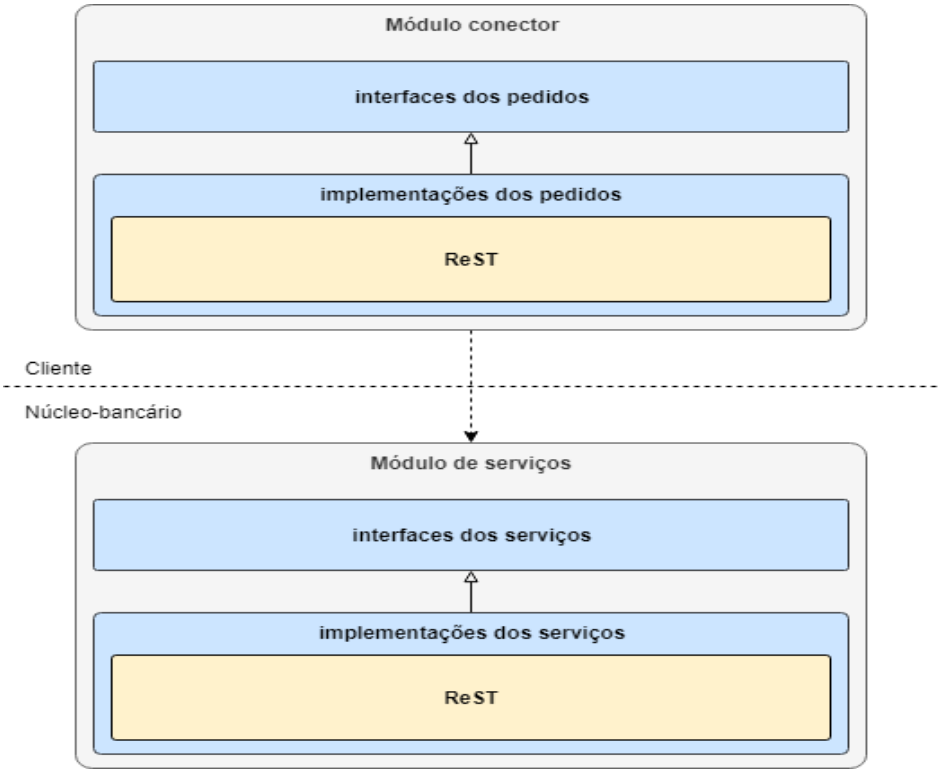


*Figura 10 - Camada de serviços detalhado*

Tal como se pode ver na Figura 10, o módulo conector disponibiliza para os seus utilizadores uma interface com os pedidos que podem ser realizados para o módulo de serviços e traduz estes pedidos através de uma implementação REST destas interfaces.

Tanto o módulo de serviços como o módulo do conector são constituídos por:

- Uma coleção de interfaces que define múltiplos contratos que devem ser honrados por quem os implementa. Mais precisamente, estas interfaces definem a forma como o cliente pode aceder às funcionalidades do núcleo, ou seja, quais os métodos que podem ser chamados e os seus parâmetros.
- Uma ou mais concretizações destas interfaces, baseadas por exemplo em serviços REST ou SOAP (43).

A decisão de fornecer estes módulos implica escrever um código ligeiramente moroso. No entanto, há o ganho de haver uma separação de conceitos entre os métodos fornecidos e as implementações e, desta forma, caso no futuro exista a necessidade de implementar uma nova forma de aceder aos serviços, apenas é necessário acrescentar um módulo dentro da subcamada de implementações e implementar as interfaces neste

novo módulo. Isto mesmo é exemplificado na Figura 11, com uma situação em que há três estilos de acesso às funcionalidades do núcleo: REST, SOAP e *remote procedure call* (RPC) (44).

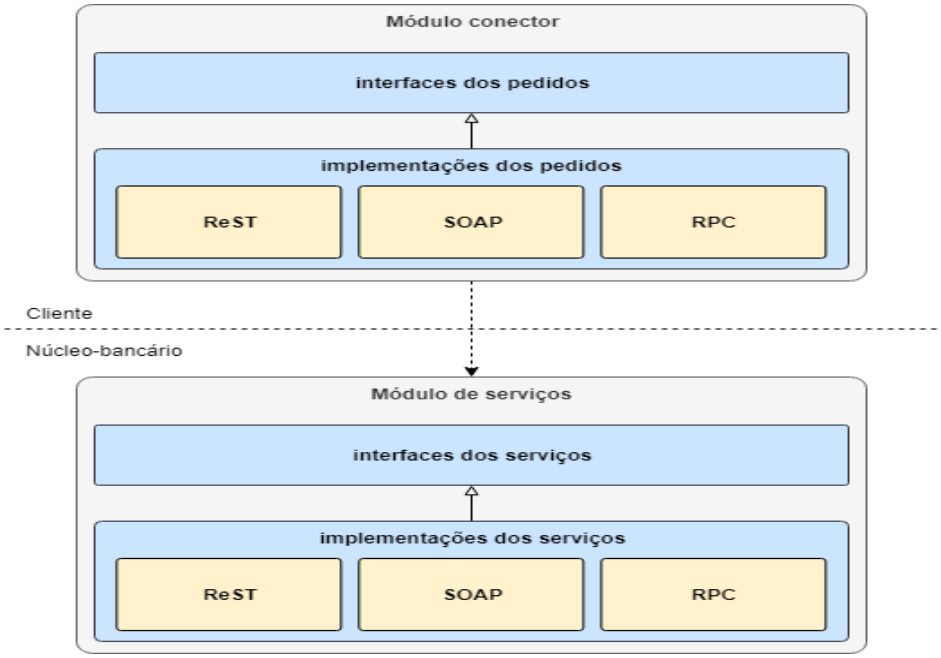


*Figura 11 - Camada de serviços detalhado com múltiplas implementações*

## 3.6 Vistas arquiteturais

Nesta secção são apresentadas algumas vistas arquiteturais focadas em diferentes estruturas de software que pretendem ajudar o leitor na comunicação sobre o software, regista as decisões mais importantes acerca do componente e permite uma compreensão geral de como o mesmo está organizado.

### 3.6.1 Estrutura de módulos – *Allowed to use*

O *núcleo-bancário* encontra-se organizado seguindo uma estrutura em módulos, tal como pode-se ver na Figura 12. Esta estrutura tem como objetivo representar unidades de código e separação de responsabilidades pelos módulos, para facilitar a manutenção e evolução do sistema. Esta separação apresenta as dependências que cada unidade tem entre si, portanto o desenvolvimento teve início nas unidades com menor número de dependências. O objetivo desta estrutura é promover a modificabilidade e portabilidade das unidades de código, facilitar a gestão da complexidade do sistema e facilitar a comunicação das unidades de código, promover a reutilização e separação de conceitos.

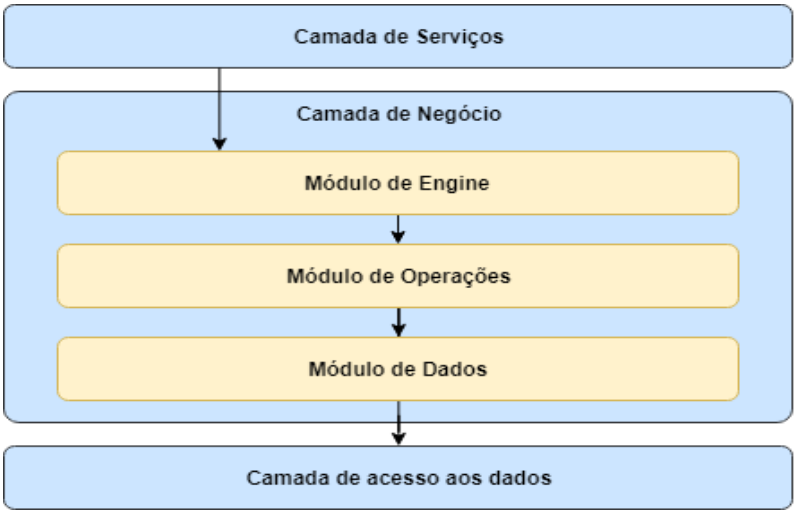


*Figura 12 - Vista de Módulos do Sistema Bancário*

***Camada de serviços***

Camada que fornece através de interfaces bem estruturadas uma forma do exterior comunicar com a camada de negócio através da internet. Neste caso em particular tira-se partido das tecnologias fornecidas pelo *package javax.ws.rs* do *Java EE*, que fornece todos os mecanismos necessários para implementar uma API REST.

***Camada de negócio***

Camada de negócio implementa a lógica do processo de negócio do núcleo. Responsável pela receção dos pedidos no módulo *engine* que então é devidamente reencaminhado para o módulo de operação que executa a logica de negócio sobre os dados e persistindo-os através da camada de acesso aos dados.

**Módulo de *engine***

O *engine* é responsável por escalonar um pedido da camada de serviços para a operação correta. Desta forma existem um *engine* para muitas operações. Estas operações estão relacionadas entre si de acordo o processo de negócio.

**Módulo de operações**

O padrão escolhido para implementar o módulo de operações é uma versão simplificada do modelo de domínio. Verificou-se apropriado para o núcleo pois é organizado em torno de operações do sistema, segue uma abordagem centrada em objetos do domínio, mas sempre com o objetivo de diminuir a complexidade.

Para evitar código repetido é tirado partido de classes gestoras (*managers*) e auxiliares (*helpers*) que ajudam a centralizar parte do código importante e com grande

probabilidade a ter erros. Desta forma aumenta-se a coesão (*cohesion*) à custa de um aumento do número de dependências (*coupling*) que as operações possuem.

Outra técnica utilizada para evitar código repetido foi criar hierarquias de classes entre operações semelhantes (exemplo: depósito e levantamento) e hierarquias entre todas as operações, já que existem determinados passos obrigatórios para praticamente todas as operações existentes no núcleo (exemplo: verificar que o cliente existe).

**Módulo de dados**

O módulo de dados foi implementado segundo o padrão repositório, onde para cada entidade do sistema existe um repositório associado que sabe como persistir e encontrar dados da base de dados. Para facilitar as interrogações feitas pelos vários repositórios, é utilizado o JPQL que é muito semelhante ao SQL, estas interrogações são realizadas sobre as entidades do domínio que são marcadas com anotações do *Java Persistence API* que é uma especificação de um mapeamento de dados (*Data Mapper)* (45). Este mapeamento implementa e abstrai a camada de acesso aos dados.

***Camada de acesso aos dados***

A camada de acesso aos dados é gerada de forma automática pelo JPA. Este tira partido das anotações que foram colocadas no módulo de dados para gerar a implementação dos vários mapas entre as entidades do domínio e os respetivos esquemas da base de dados.

Esta camada é completamente transparente para o desenvolvedor, único passo necessário é colocar as anotações corretamente no módulo de dados.

### 3.6.2 Estrutura de componentes – Cliente servidor

Objetivo desta estrutura é representar as principais unidades de computação e repositórios de dados que existem em tempo de execução e nos meios de comunicação usados de forma a realizar as funcionalidades do sistema. A Figura 13 Apresenta quais são os componentes principais e como estes interagem entre si através dos conectores e protocolos associados.

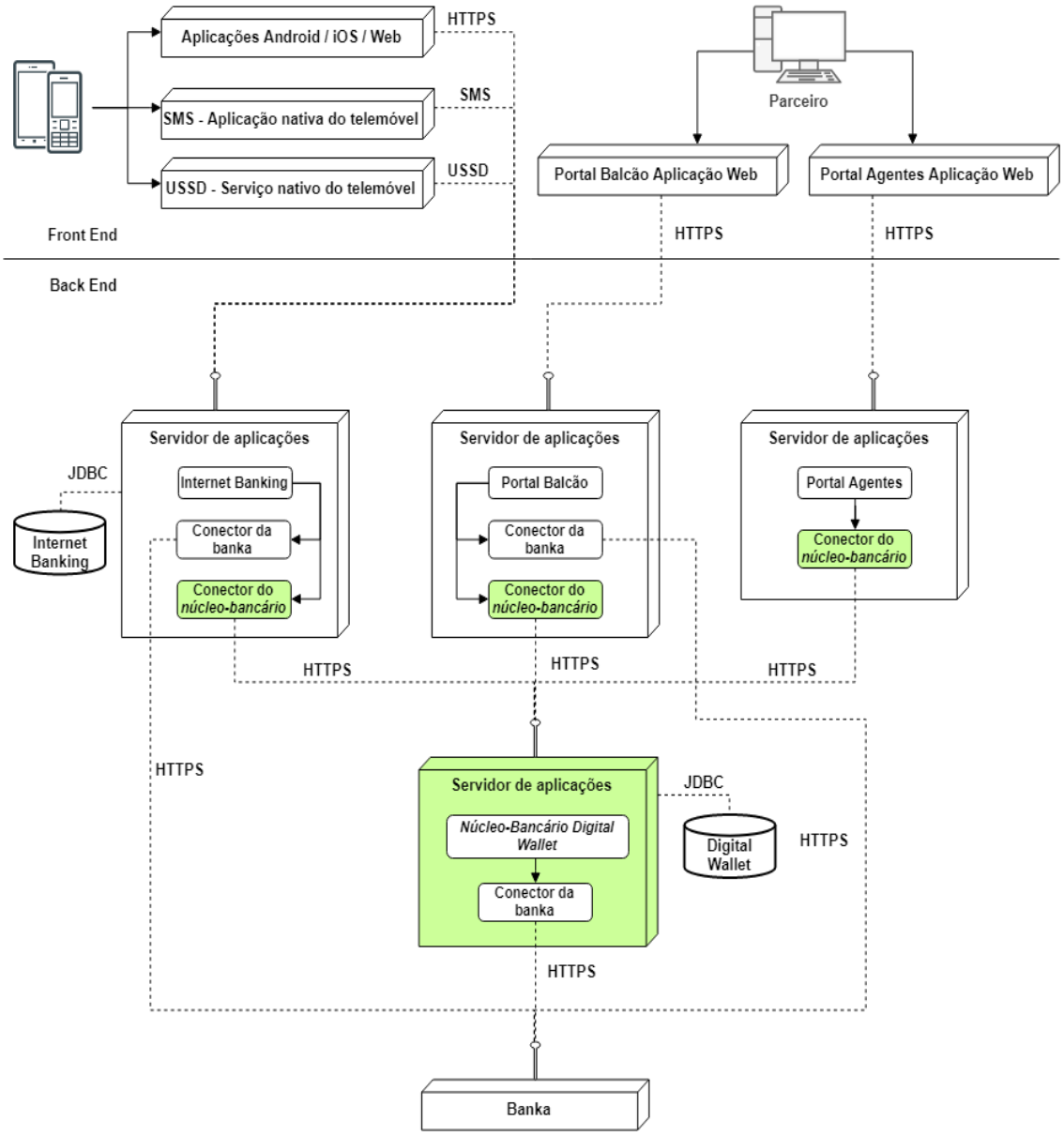


*Figura 13 - Vista de Componentes e Conectores do Sistema Bancário*

## *Componentes*

### Cliente

Identifica um cliente do sistema. o cliente tira partido das aplicações Android, iOS e Web para gerir seus dados e suas contas.

### Aplicações Android / iOS / Web

Componente que representa as principais aplicações disponibilizadas pelo sistema bancário. Fornecem ao cliente um meio de aceder, via internet, os seus movimentos bancários, gerir suas contas, gerir os seus dados e executar transferências, entre outras operações. No contexto do sistema *Digital Wallet*, estas aplicações apresentam ao

cliente as funcionalidades de criar / gerir uma nova conta *wallet* e realizar movimentações sobre estas.

**SMS – aplicação do cliente**

Componente que representa uma aplicação de SMS no telemóvel do cliente. No contexto bancário é possível enviar uma mensagem de texto para o servidor Internet Banking e receber resultados na forma de uma mensagem de texto. Devido a pouca segurança deste sistema, é apenas possível realizar consultas com uma grande limitação. A autenticação deste canal baseia-se no número de telemóvel do cliente.

**USSD – aplicação do cliente**

Componente que representa uma aplicação USSD no telemóvel do cliente. Através do canal USSD é possível realizar consultas limitadas sobre as contas do cliente, principalmente devido a falta de segurança associada a estes canais. No entanto, será possível executar transferências associadas a contas *wallet* com a autenticação de segundo nível através de um pin gerado na criação do cliente. Este pin deve ser conhecido apenas pelo cliente.

**Parceiro**

Identifica um parceiro do sistema. Os parceiros tiram partido das aplicações web Portal Balcão ou Portal Agentes para parceiros do tipo operador ou agentes respetivamente.

**Portal Balcão aplicação web**

Aplicação utilizada pelos operadores de caixa dos balcões da instituição financeira, que fornece as funcionalidades relacionadas com a gestão de clientes, das suas contas e deposito e levantamentos de numerário.

**Portal Agentes aplicação web**

Aplicação utilizada pelos agentes, que fornece as funcionalidades relacionadas com a gestão de clientes e das suas contas.

**Internet Banking web server**

Servidor Web que fornece as funcionalidades das aplicações Android, Ios, Web, SMS e USSD para os clientes.

**Portal Balcão web server**

Servidor Web que fornece as funcionalidades da aplicação Portal Balcão.

**Portal Agentes web server**

Servidor Web que fornece as funcionalidades da aplicação Portal Agentes.

***Núcleo-bancário Digital Wallet***

Servidor Web do núcleo-bancário *Digital Wallet* que fornece as funcionalidades para os restantes servidores de aplicações do sistema.

**Digital Wallet Connector**

Implementação dos pedidos REST do componente *núcleo-bancário Digital Wallet* do lado do cliente. Cuja função e facilitar e aumentar a velocidade de integração dos outros componentes.

**Banka**

Componente aplicacional que faz a gestão das movimentações bancária das contas dos clientes. Este componente encontra-se interno a instituição financeira e, portanto, inclui as regras de negócio especificas da instituição e regras obrigatórias no país para transações bancárias.

**Banka Connector**

Implementação dos pedidos REST do componente Banka do lado do cliente. Cuja função é facilitar e aumentar a velocidade de integração dos outros componentes.

**Base de dados – Internet Banking**

Base de dados que armazena apenas os dados do componente Internet Banking.

**Base de dados – Digital Wallet**

Base de dados que armazena apenas os dados do componente *núcleo-báncario Digital Wallet*.

***Conectores***

**Conector HTTPS**

Conector utilizado para transmitir dados entre componentes através dos protocolos HTTP de forma segura, tira-se para isto partido do protocolo TLS (46). A principal motivação para o uso do HTTPS é a autenticação do servidor perante o cliente e proteger a comunicação através de cifras a manter a privacidade e integridade dos dados transmitidos.

**Conector SMS**

Conector utilizado para transmitir uma mensagem SMS entre a aplicação de SMS do cliente e o servidor web Internet Banking.

**Conector USSD**

Conector utilizado para transmitir e manter uma sessão USSD entre o cliente e o servidor web Internet Banking.

**Conector JDBC**

Conector utilizado para conectar os servidores e executarem interrogações com a base de dados.

## 3.7 Ambientes

Durante o processo de engenharia do *software* do núcleo-bancário, o *software* passa por três ambientes: ambiente de desenvolvimento, ambiente de qualidade e ambiente de produção. Cada ambiente tem como objetivo testar o software de forma a que seja encontrado o maior número de erros antes deste ser publicado aos clientes finais.

### 3.7.1 Ambiente de desenvolvimento

Ambiente mais utilizado durante todo o processo de criação do software, onde deve ser detetado o maior número de erros existentes no *software* em criação. Este ambiente é um ambiente controlado, ou seja, é um ambiente interno a empresa acolhedora onde tudo o que acontece nas máquinas deste ambiente tem origem interna, desta forma apenas os *softwares* desenvolvidos ou os testes feitos pela equipa de testes interna, podem afetar de alguma forma o software e/ou as máquinas.

Neste ambiente todos os dados são fictícios de forma a serem testados sem restrições pelas equipas de desenvolvimento e equipas de testes.

### 3.7.2 Ambiente de qualidade

Tal como no ambiente de desenvolvimento, este ambiente tem como objetivo testar novamente o *software* em todas as suas funcionalidades. Distingue-se do ambiente de desenvolvimento em duas frentes.

A primeira diferença do ambiente de desenvolvimento é o facto das máquinas onde o servidor estar alojado são máquinas internas à instituição financeira. Estas máquinas são semelhantes as de produção. Onde o objetivo e ser o mais próximo do ambiente real possível.

A segunda diferença é o facto deste ambiente ser apenas utilizado após algumas iterações no ambiente de desenvolvimento. Isto implica que há menos substituições do software em execução, o que implica que este ambiente é mais estável e capaz de ser testado tanto pela equipa de testes internas da empresa acolhedora, como pelo cliente do *software*.

### 3.7.3 Ambiente de produção

Em ambiente de produção o núcleo-bancário terá duas fases: A fase de testes em produção e fase de lançamento. Em termos de máquinas este ambiente é igual ao ambiente de qualidade, distinguindo-se apenas em relação aos dados, que neste caso, são dados reais com movimentações reais de dinheiro. Neste caso o processo de instalação do ambiente em produção deve se iniciar com testes em produção de contas reais, mas onde as movimentações pertencem aos clientes criados apenas com o propósito de testes de produção para a empresa acolhedora e clientes criados para testes da instituição financeira.

De seguida inicia-se a fase dois com o lançamento do software para o publico alvo.

## 3.8 Verificação e validação do software

Nesta secção pretende-se apresentar testes efetuados sobre o núcleo-bancário de maneira a validar junto com os testes funcionais que o núcleo implementa o negócio pretendido de acordo os requisitos levantados nas etapas anteriores do processo de engenharia. Apresenta-se também testes efetuados a nível de desempenho do núcleo em tempo de execução em ambiente de desenvolvimento interno à empresa acolhedora.

### 3.8.1 Testes funcionais

Devido à modularidade do sistema, foi possível testar seus módulos de forma simples e rápida. Para tal tiramos partido da ferramenta JUnit4 (47) que é uma plataforma especializada para testes do código desenvolvido. Com o JUnit4 é possível escrever pequenas funções que verifica o código desenvolvido no núcleo em relação aos requisitos. Os testes devem ser independentes uns dos outros e devem verificar uma determinada funcionalidade. Em caso de erro, pode-se identificar, com maior precisão, o local onde a origem do erro se encontra.

Os testes funcionais foram implementados e executados sobre dois módulos principais:

- Módulo de dados implementa os repositórios que possuem as interrogações em JPQL sobre as entidades do domínio. Existe assim para cada entidade do domínio um repositório e consequentemente existe um teste associado à entidade em causa.

Por exemplo para a entidade *Client* que representa um cliente no sistema, existe um repositório *ClientRepository* e um *ClientRepositoryTest.* Sobre o repositório é possível realizar operações de criar, atualizar, ler e eliminar entidades.

- Módulo de operações implementa as operações disponibilizadas pelo *núcleo-bancário* e mais especificamente pelo *engine*. Cada operação é uma classe java e corresponde à uma classe de testes. Por exemplo para a operação criar cliente existe a classe que implementa a operação *CreateNewClientEngineOperation* e existe uma classe de testes que implementa os testes sobre esta operação a *CreateNewClientEngineOperationTest*.

Os problemas aparecem quando há dependências entre módulos ou dependências de outras bibliotecas. Para tal utiliza-se o Mockito (48) que permite criar implementações falsas que imitam as dependências. Pode-se assim garantir que as funcionalidades sob testes estão corretas, ou em caso de erros, garante que os erros não têm origem nas dependências.

Os testes ao módulo de dados visam garantir que as interrogações JPQL e as anotações JPA estão corretas. Desta forma tem-se confiança na estabilidade do sistema e na integridade dos dados, por exemplo: caso um parâmetro esteja marcado como não nulo com anotações JPA, testa-se a criação de entidades com este campo a nulo e verifica-se que não é possível. Estes testes são simples e rápidos de serem feitos. Sendo que apenas é preciso testar sobre os repositórios. A única dependência deste modulo é o *entity manager* do JPA, o *entity manager* é o meio utilizado pelo repositório para aceder e persistir dados. Para que todos os testes executados no sistema garantissem os mesmos resultados, independente de quantas vezes estes fossem executados, utiliza-se uma base de dados em memória, que é criada no início dos testes, limpa entre testes, e destruída no final.

Os restantes testes focam-se sobre o módulo de operações. Este módulo tem como principal dependência o módulo de dados. É nesta situação que é possível tirar proveito do Mockito, cria-se classes falsas dos repositórios que são injetadas nas operações sob teste. Estas classes falsas permitem retornar valores falsos nas chamadas dos métodos e validar de forma mais consciente e correta a validade da operação sob teste.

Não existem testes sobre a camada de serviços REST, devido a sua natureza pobre de lógica, sendo que as validações dos parâmetros de entrada são feitas na camada de negócio. A camada de serviços apenas serve para receber e reencaminhar os pedidos para a camada inferior, a camada de negócio.

### 3.8.2 Testes de desempenho

Para além dos testes funcionais, foram realizados testes de desempenho do sistema com testes de carga sobre o *núcleo-bancário*. Os testes de desempenho foram efetuados num ambiente de desenvolvimento interno à empresa acolhedora, onde o objetivo era analisar o comportamento e estabilidade do sistema diante de um determinado número de pedidos feitos em paralelo.

Os testes de desempenho foram feitos com a ferramenta Apache JMeter™ (49). Esta ferramenta permite uma elevada configuração da maneira como é feito os pedidos ao servidor sob testes.

Para efetuar os testes foi preciso seguir o seguinte procedimento:

1. Instalar o software do núcleo-bancário desenvolvido num servidor de aplicações. Este servidor encontra-se instalado nas máquinas servidoras da entidade acolhedora, que para efeitos de desenvolvimento, era partilhado com outras equipas. Isto revela-se importante, pois estes podem influenciar de forma negativa os testes de desempenho.
2. Ligar a máquina que irá executar a ferramenta Apache JMeter™ à rede interna da empresa acolhedora, cujo objetivo é diminuir a latência do pedido na rede. Utiliza-se duas máquinas com o propósito de não partilhar tempo de processamento para fazer pedidos e para processar o pedido.
3. De seguida configurar o JMeter de acordo com o plano de teste a ser executado. A priori configura-se uma sequência de pedidos que irá ser executada pelos clientes. Entre testes configura-se o número de clientes – número de *threads* – que serão lançados e o número de vezes que a sequência de pedidos será executada. Por exemplo: Lançar 2 clientes que irão executar 2 vezes uma sequência constituída por 5 pedidos. Implica que haverá 2 pedidos a serem feitos ao servidor em simultâneo e que serão feitos 20 pedidos. Cada cliente irá executar 10 pedidos.

A máquina do servidor onde o núcleo foi testado possui as seguintes especificações: Cpu AMD Opteron™ Processor 6238 com 2593.495 MHz e 4 núcleos, 4GBs de memória Ram.

O JMeter™ fornece a funcionalidade de realizar pedidos ordenados a interface de serviços disponibilizada pelo núcleo. Este tipo de testes visa representar uma sequência de chamadas que um utilizador do sistema poderia vir a efetuar em uma utilização normal. Consiste em realizar, por exemplo, o carregamento com dinheiro da conta antes de tentar fazer uma transação.

Ao processar os dados obtidos com o JMeter™ foi possível construir a Tabela 1. Esta tabela apresenta o número médio de pedidos que o servidor consegue suportar dado

um número de clientes a fazerem pedidos em simultâneo. Quer isto dizer que existem em paralelo um número x de clientes a executarem uma sequência de pedidos ao servidor e no fim de cada pedido é inicializado o próximo até atingir os 9000 pedidos no total. 9000 pedidos foi o número escolhido para representar uma carga suficientemente elevada para obter dados significativos e pequeno o suficiente para ser executado em tempo útil.

O *throughput* representa a taxa de processamento que, em teoria, o servidor consegue suportar por segundo, ou seja, o número de pedidos que o servidor consegue processar a cada instante. Este *throughput* é dado pela fórmula T.

$$T = \ número\ total\ de\ pedidos\ /\ duração\ total\ do\ teste$$

| Número de clientes | Número total de pedidos | Tempo médio de todos os pedidos [ms] | Percentagem de Erros | Duração total do teste [ms] | *Throughput* [pedidos/s] | Desvio [ms] |
|---|---|---|---|---|---|---|
| 1 | 9,000 | 245 | 0.00 | 2,220,000 | 4.1 | 233 |
| 100 | 9,000 | 11,685 | 0.00 | 1,060,000 | 8.5 | 8,396 |
| 300 | 9,000 | 39,709 | 0.00 | 1,193,000 | 7.5 | 32,860 |
| 600 | 9,000 | 89,483 | 18.44 | 1,352,000 | 6.7 | 64,267 |
| 1000 | 9,000 | 153,253 | 61.86 | 1,427,000 | 6.3 | 49,613 |

*Tabela 1 - Resultados do Apache JMeter™*

Os testes de desempenho foram feitos em 5 recolhas de 9000 pedidos cada, é possível verificar que a duração do tempo de resposta do serviço aumenta de maneira linear proporcional ao número de clientes, como pode-se verificar na Figura 14. Esta realidade deve-se ao facto de existir um *bottleneck* no sistema. Este *bottleneck* pode ter origem nas configurações da máquina, como por exemplo a quantidade de memória disponível; como pode ter origem no acesso à base de dados. Este *bottleneck* apresenta um maior impacto no desvio da duração da resposta que existe e que é cada vez maior quando o número de clientes aumenta. Este desvio é menor no último caso, devido ao elevado número de pedidos, 61.86%, que deram erro por *timeout*.

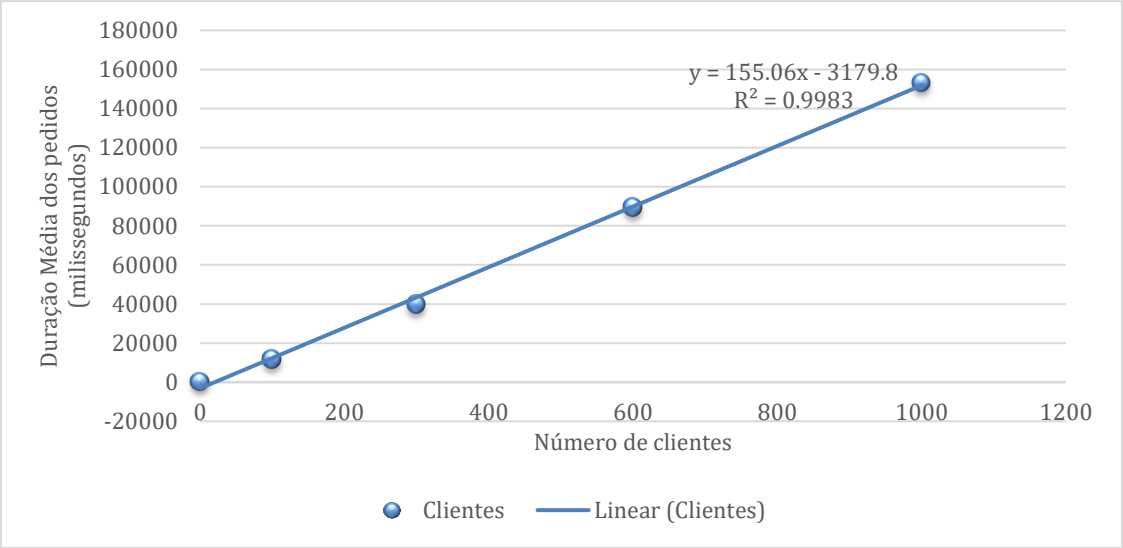


*Figura 14 - Duração Média dos pedidos dado o número de clientes em simultâneo*

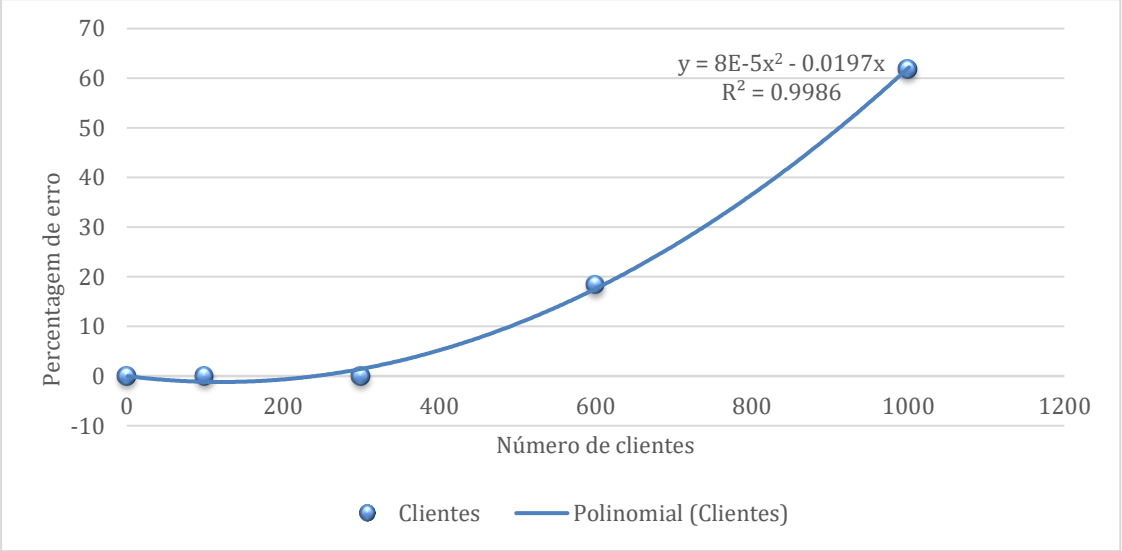


*Figura 15 - Percentagem de erro dado o número de clientes em simultâneo*

Dado os dados apresentados na Tabela 1 foi possível traçar dois gráficos: Figura 14 e Figura 15, e respetivos gráficos f(x) e g(x) onde x representa o número de clientes a acederem o servidor em simultâneo.

$$f(x) = 155.06x - 3179.8$$

$$g(x) = 0.00008x^2 - 0.0197x$$

Desta forma é possível concluir que com o estado atual do servidor e da base de dados o número ideal de clientes em simultâneo para uma determinada taxa de erro implica um tempo máximo de resposta igual ao apresentado na Tabela 2.

| Percentagem de erro máxima | Número de clientes máximo | Tempo máximo de resposta esperado [ms] |
|---|---|---|
| 10 | 497 | 73,885 |
| 5 | 401 | 58,999 |
| 1 | 289 | 41,632 |
| 0.5 | 269 | 38,531 |
| 0.1 | 251 | 35,740 |
| 0 | 246 | 34,964 |

*Tabela 2 - Taxa de erro máximo dado um número de clientes em simultâneo*

A execução de testes não é só necessária, mas essencial a qualquer *software* para garantir que este corresponde as funcionalidades de negócio - através de testes funcionais – e testes de desempenho para verificar a estabilidade do software em tempo de execução.

Através da implementação e execução de testes foi possível verificar o estado atual do sistema e identificar problemas que não seriam possíveis identificar sem os testes efetuados, nomeadamente:

- Os testes funcionais permitiram identificar erros a nível de codificação das operações e de maneira célere foi possível corrigir pequenos erros antes de instalar nas máquinas para integração com os outros componentes.
- Os testes de desempenho permitiram identificar um *bottleneck* no sistema, que não poderia ser identificado de outra forma a não ser através destes testes. Através do conhecimento obtido, é possível, investigar e melhorar partes do sistema de forma a aumentar o desempenho do mesmo. Como por exemplo um balanceador de carga e cache de pedidos.

## 3.9 Conclusão

Este capítulo mostra de forma mais detalhada a solução proposta e desenvolvida durante este projeto de mestrado.

O processo utilizado para o desenvolvimento do núcleo bancário foi então uma metodologia que segue o modelo incremental. É necessário evidenciar que há múltiplas equipas envolvidas no desenvolvimento do sistema *Digital Wallet*, tais como as equipas responsáveis pela manutenção do Internet Banking e do Portal Balcão. No entanto o núcleo-bancário foi desenvolvido na sua totalidade pelo aluno deste mestrado, sendo auxiliado pelo supervisor durante todas as etapas do processo de engenharia.

São apresentados os casos de usos que o sistema deve suportar e é apresentado uma visão global do sistema de forma a melhor compreender como os múltiplos elementos que compõem o sistema se relacionam e como este comunicam entre si.

Este capítulo encerra-se por apresentar os múltiplos ambientes de desenvolvimento em que o núcleo esteve presente. Durante a escrita deste relatório, o núcleo encontra-se em ambiente de produção interno a instituição financeira, na etapa de testes em produção, antes de ser disponibilizado ao público.

# Capítulo 4 Sistema de pagamentos interoperável

Neste capítulo é apresentado o sistema de pagamentos interoperável. O capítulo inicia-se com a definição do problema e os requisitos que a solução deve endereçar. Mostra-se porque duas soluções arquiteturais populares não são adequadas para o sistema de pagamentos.

O capítulo apresenta ainda o conceito de *ledger* distribuído e as principais tecnologias utilizadas para desenvolver novos *ledgers* e termina com a descrição da solução escolhida e implementada.

### 4.1.1 Problema

Atualmente existe uma miríade de métodos diferentes de pagamentos com particularidades distinta entre estes. No caso de um pagamento entre duas entidades distintas, a transferência poderá não ser realizada em tempo real, sendo esta colocada numa fila de transações que posteriormente irá ser compensada e reconciliada entre as várias entidades envolvidas na transação. Além da demora associada à reconciliação financeira entre as entidades de pagamento, existem as taxas que as instituições financeiras cobram sobre as movimentações. Estas taxas aumentam consoante a celeridade do pagamento – quanto mais rápido mais caro – e com o número de instituições financeiras intermediárias que cobram uma taxa sobre a movimentação.

Face a estas dificuldades, existe assim a necessidade de substituir o sistema de pagamentos atual por um que faça as compensações financeiras de forma final, em tempo real e a baixo custo.

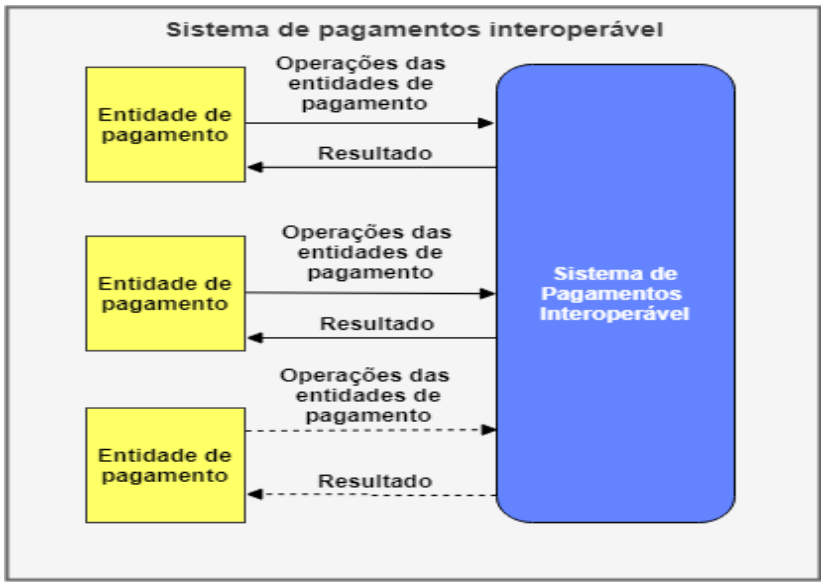


*Figura 16 - Diagrama de contexto do sistema de pagamentos interoperável*

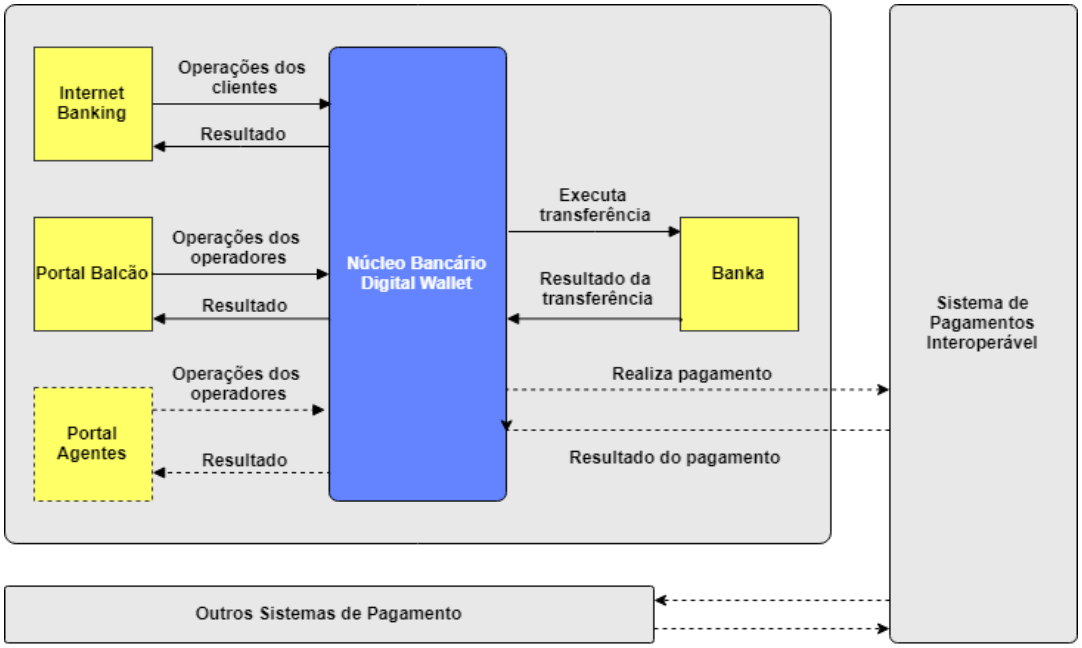


*Figura 17 - Diagrama de contexto do núcleo bancário digital wallet integrado com o sistema de pagamentos interoperável*

A Figura 16 apresenta o diagrama de contexto do sistema de pagamentos interoperável. As entidades de pagamentos devem chamar sobre o sistema as operações para realizar ou solicitarem pagamentos. Eventualmente, pretende-se evoluir o *núcleo-bancário* e integrar o sistema de pagamentos interoperável, aumentando desta forma a celeridade dos pagamentos feitos através da utilização de contas *wallets*. A Figura 17, mostra a integração do sistema de pagamentos com a *Digital Wallet*.

## 4.2 Requisitos da solução

Nesta secção pretende-se apresentar as restrições e requisitos da solução. Começa por apresentar os requisitos não funcionais para a solução que envolve a descrição da forma como o sistema deve se comportar e como este deve ser estruturado de forma a responder ao problema descrito. Termina por apresentar os requisitos funcionais e um diagrama do domínio do problema para facilitar a compreensão dos conceitos mais importantes que a solução deve endereçar.

### 4.2.1 Restrições e requisitos não funcionais

Foram identificados as seguintes restrições e requisitos não funcionais para este sistema:

- O sistema deve ser descentralizado, não necessitando que exista uma entidade reguladora intermediária que tenha controlo sobre a rede.
- O sistema deve permitir facilmente a entrada de novos nós. Esta entrada deve ser controlada e gerida pelos administradores da rede. Desta forma garante-se que apenas entidades válidas podem entrar no sistema.
- O sistema deve ter mecanismos de aceitação de novos pagamentos de uma entidade para outra entidade da rede e garantir que este pagamento é realizado em tempo útil, havendo a reconciliação bancária no momento do pedido.
- O sistema deve ter mecanismos que garantam a integridade dos dados armazenados, não podendo haver atualizações dos dados depois destes já terem sido armazenados (caso contrário seria possível fazer um pagamento de dez euros e atualizar para cem).
- O sistema deve ter mecanismos para requisitar um pagamento de outra entidade para que este pagamento possa vir a ser pago no futuro. Deve existir um tempo limite entre a requisição e a execução deste pagamento. Caso este limite seja ultrapassado, a solicitação deve expirar.

A decisão foi criar um sistema que tira partido das tecnologias de *ledgers* distribuídos, mais especificamente da tecnologia *blockchain*. A natureza descentralizada deste tipo de registos permite resolver uma grande parcela dos requisitos.

As entidades de pagamento necessitam enviar e receber dinheiro para que exista a compensação financeira entre ambas as partes. Essa movimentação, em princípio, ocorre com a transferência de dinheiro entre contas aprovisionadas em instituições financeiras – o que leva a atrasos e taxas elevadas. De forma a ultrapassar esta dificuldade, o sistema irá usar *tokens*. O *token* representa um ativo financeiro digital interno ao sistema, que as entidades aceitam como forma de troca de valores.

### 4.2.2 Requisitos Funcionais

Relativamente aos requisitos funcionais, foi feito um levantamento dos casos de uso que o sistema de pagamentos interoperável devia suportar. Estes casos de uso devem ser genéricos o suficiente para suportar múltiplos tipos de entidades de pagamento.

***Domínio***

Para ajudar a compreensão dos requisitos funcionais, esta subsecção se inicia por apresentar um diagrama do domínio (Figura 18) e uma descrição dos conceitos da solução.

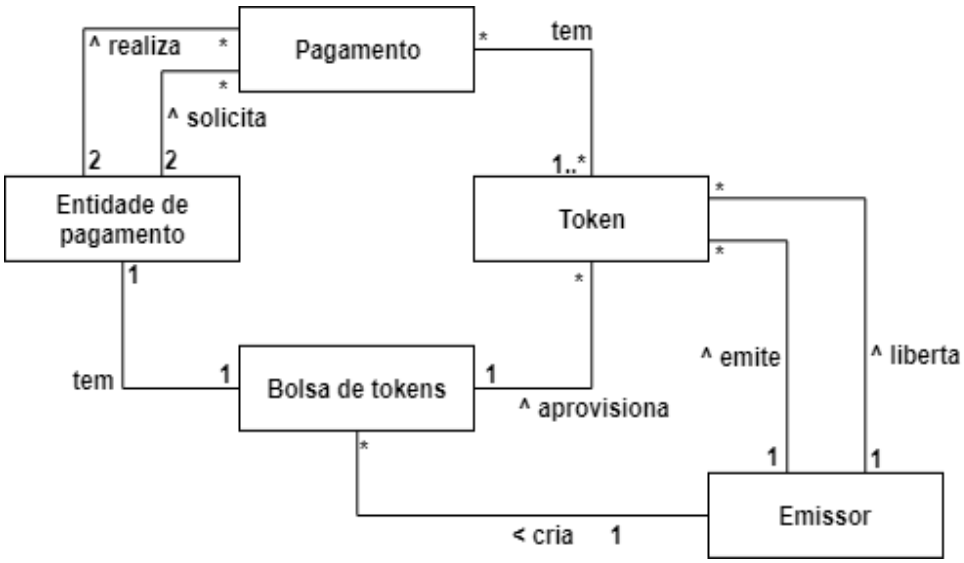


*Figura 18 - Diagrama modelo de domínio do sistema de pagamentos interoperável*

**Conceitos**

*Entidade de pagamentos*

Representação de uma entidade de pagamento no sistema de pagamentos. Cada entidade pode realizar pagamentos para outras entidades ou pode solicitar pagamentos a outras entidades. Associado a cada entidade existe uma bolsa de *tokens*.

*Bolsa de tokens*

A bolsa de *tokens* representa o local que armazena os *tokens* que uma determinada entidade de pagamentos possui. Para poder realizar ou solicitar um pagamento, a entidade deve ter uma bolsa de *tokens.*

*Pagamento*

No sistema, um pagamento consiste na transferência de um montante entre duas entidades. Mais concretamente, consiste na retirada de *tokens* da bolsa de *tokens* da entidade origem e o depósito de *tokens* na bolsa da entidade destino.

*Token*

O *token* é o ativo financeiro nativo do sistema que e utilizado para troca de valores entre entidades que participam num pagamento.

*Emissor*

Para poder efetuar os pagamentos, as entidades necessitam da bolsa de *tokens* e dos *tokens.* A criação e emissão de *tokens* é feita através do emissor. O emissor tem como função criar a bolsa com saldo zero para as entidades de pagamentos quando esta entra no sistema.

A entrada de novos *tokens* no sistema também é realizada através do emissor, ou seja, este tem o papel de converter dinheiro em *tokens* do sistema. O emissor também desempenha o papel de libertar *tokens* do sistema. Neste caso há a saída de *tokens* do sistema e a sua consequente conversão em dinheiro.

**Restrições**

Estes conceitos, e a forma como se relacionam, estão sujeitos às seguintes restrições:

- Os *tokens* utilizados para realizar um pagamento feito por uma entidade, deve sair da bolsa de *tokens* desta entidade.
- Os *tokens* utilizados para solicitar um pagamento para uma entidade, deve sair da bolsa de *tokens* da entidade solicitada e entrar na bolsa de tokens da entidade que solicita.
- O emissor emite *tokens* para uma bolsa de *tokens*.
- O emissor liberta *tokens* de uma bolsa de *tokens*.
- A emissão ou libertação só podem ser efetuadas sobre montantes positivos de *tokens*.
- A quantidade de *tokens* em uma bolsa não pode ser menor que zero.
- Não é possível efetuar pagamentos de montantes inferiores a zero.
- Não é possível solicitar pagamentos de montantes inferiores a zero.
- Uma entidade de pagamento só pode realizar ou solicitar pagamentos para outras entidades.

### *Casos de uso*

As operações do sistema interoperável de pagamentos são apresentadas de forma visual na Figura 19. Estas operações devem ser simples e genéricas o suficiente para poder suportar pagamentos provenientes de entidades de pagamentos com sistemas distintos entre si.

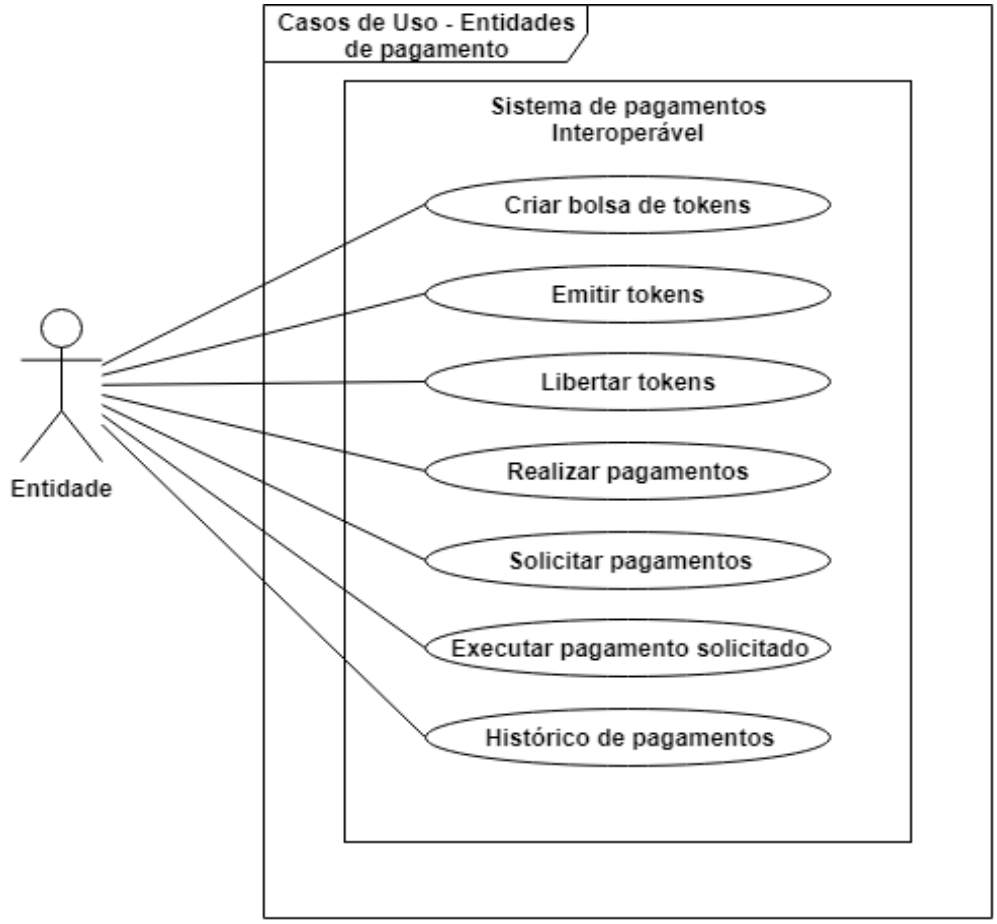


*Figura 19 - Casos de uso das entidades de pagamento*

**Criar bolsa de tokens**

Caso de uso que permite criar uma bolsa de *tokens* com saldo zero para uma entidade de pagamentos.

**Emitir | Libertar tokens**

Dado um montante e uma entidade, este caso de uso permite emitir novos *tokens* ou libertar *tokens* de uma bolsa associada a uma entidade de pagamentos.

**Realizar pagamento**

Dado uma entidade destino e um montante, este caso de uso permite executar a transferência de um montante em *tokens* da bolsa da entidade origem para a bolsa da entidade destino, havendo assim a reconciliação financeira e finalização.

**Solicitar pagamento**

Ao contrário do anterior, neste caso dado uma entidade de origem e um montante, permite solicitar um pagamento para ser feito pela entidade origem para a entidade destino, que neste caso é a entidade que solicita o pagamento. O pagamento fica com um estado de solicitado e neste exato momento não existe reconciliação nem finalização do pagamento.

**Executar pagamento solicitado**

Associado à solicitação de um pagamento (caso de uso anterior) este caso de uso deve ser executado pela entidade cujo pagamento foi solicitado. É verificado se o pagamento é válido e então é executada a reconciliação e finalização do pagamento. Há neste exato momento a transferência de *tokens* entre as bolsas de ambas as entidades.

**Histórico de pagamentos**

A entidade deve ter a capacidade de consultar o histórico de pagamentos executados onde esta esteve envolvida.

## 4.3 Análise de duas soluções tradicionais

Nesta secção analisam-se duas soluções alternativas para a arquitetura do sistema de pagamentos que são baseadas em estilos tradicionais em sistemas distribuídos.

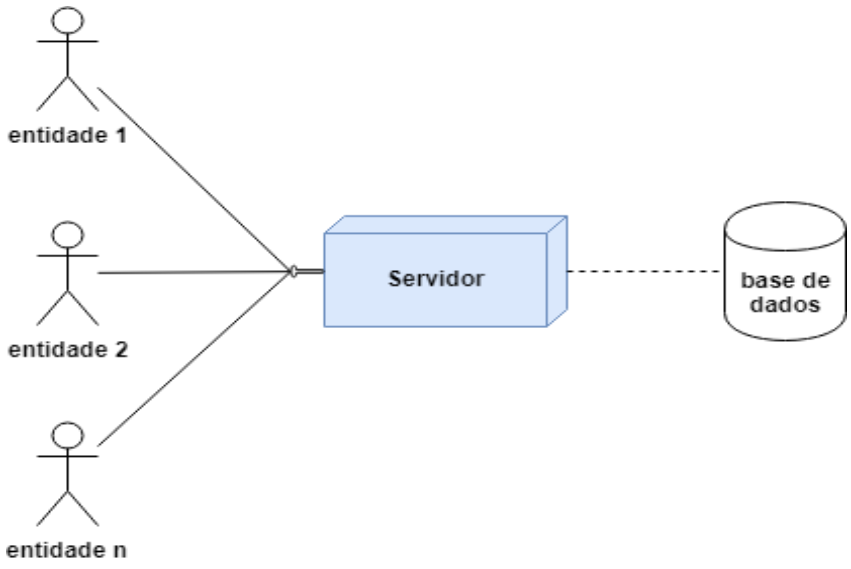


*Figura 20 - Alternativa 1 do sistema de pagamentos*

**Alternativa 1**: Nesta alternativa, que segue um estilo tradicional de sistemas distribuídos, existem n entidades de pagamento, um servidor e uma base de dados, organizados em *3-tiers* como mostrado na Figura 20.

Na Figura 20 é possível ver que nesta alternativa as entidades de pagamentos são clientes do sistema, e não participam de forma ativa na escrita e leitura dos dados. É possível ver que existe uma entidade central – o servidor – que é responsável pela gestão das transações e da escrita e da leitura dos dados da base de dados. Existe uma base de dados relacional com várias tabelas que representam as entidades, os pagamentos e possivelmente outros dados relevantes. Esta base de dados poderia e deveria ser replicada de forma a manter a confiança e disponibilidade dos dados.

**Análise:** Esta solução alternativa tem dois problemas principais:

- Existência de uma entidade central, o servidor, que além de ser um ponto único de falha, é unicamente responsável pela rede. Isto implica que este servidor pode cobrar taxas elevadas sobre as transações, sendo este um dos problemas que se pretende solucionar.
- A possibilidade de haver atualizações dos dados na base de dados. O sistema de pagamentos não deve permitir este tipo de operações sobre os dados. Sendo este, novamente, um dos problemas que se pretende resolver.

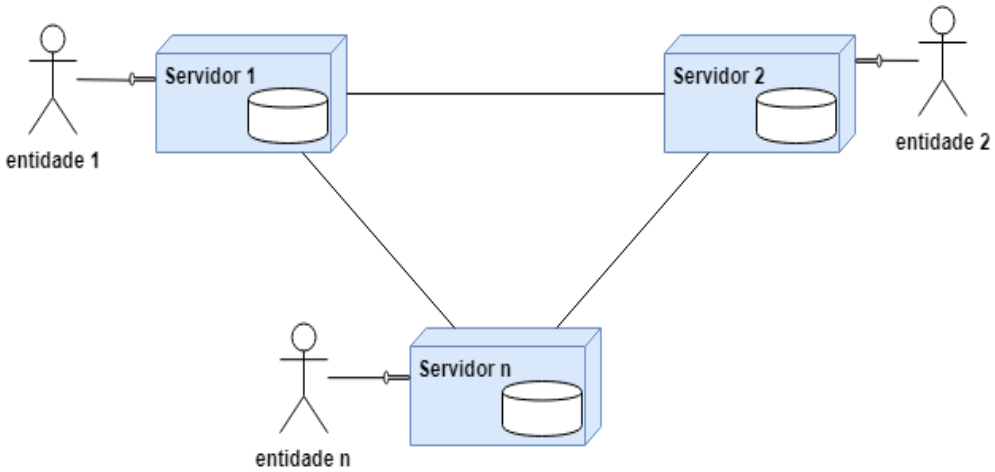


*Figura 21 – Alternativa 2 do sistema de pagamentos*

**Altenativa 2**: Esta alternativa passa por ter uma rede de nós onde agora as entidades de pagamento são representadas como um nó, composto por um servidor e uma base de dados interna, tal como é ilustrado na Figura 21. Esta nova arquitetura é distribuída e descentralizada. No entanto, sendo estas bases de dados "tradicionais", é muito fácil alterar os dados das entradas e inviabilizar o sistema com dados distintos entre as bases de dados ou com valores que não representam o que realmente aconteceu.

O problema da atualização dos dados poderia ser mitigado com o auxílio de funções de dispersão, como exemplificado na Tabela 3.

| Id | Entidade origem | Entidade destino | Montante | *Hash* da linha anterior | *Hash* da linha atual |
|---|---|---|---|---|---|
| 1 | A | B | 10 | Abc123f | Ffg456g |
| 2 | A | C | 15 | Ffg456g | Ggh789h |

*Tabela 3 - Caso de estudo 2: tabela da base de dados*

A Tabela 3 mostra o que poderiam ser duas linhas de uma tabela de forma a manter a integridade dos dados. Nesta situação cada linha armazena o *hash* da linha anterior e gera o seu *hash*. Esta solução impossibilita a alteração dos dados da base de dados a menos que se altere toda a tabela. Existe apenas um ponto de falha que é o último elemento: como o seu *hash* ainda não se encontra inserido em outra linha é possível ser alterado. Por outro lado, este tipo de abordagem aumentaria a complexidade do sistema de forma desnecessária, tendo em conta que já existe uma estrutura de dados que simplifica e tem como premissa esta abordagem, a *blockchain.*

## 4.4 *Ledger* distribuídos

Um *ledger* distribuído é uma base de dados descentralizada e distribuída que permite que diferentes entidades que não confiam totalmente umas nas outras mantenham um consenso sobre a existência, o estatuto e a evolução de um conjunto de factos partilhados. Frequentemente, isto pode traduzir-se pelo consenso relativamente à existência de um conjunto de transações. Neste caso, a introdução de uma nova transação no *ledger* exige a aceitação desta pelos validadores em que o sistema confia para realizar a verificação e validação das informações presentes na transação em causa e se esta pode entrar no *ledger* do sistema. Esta validação é feita através de um protocolo de consenso que irá ser discutido mais à frente.

As transações podem ser armazenadas recorrendo a diferentes estruturas, materializando em diferentes tecnologias que dão suporte a *ledger* distribuídos. Para este trabalho considerou-se a tecnologia de *blockchains*, a qual usa uma cadeia de blocos para armazenar as transações, por ser a tecnologia atualmente predominante neste género de base de dados.

### 4.4.1 Como funciona

O objetivo do *ledger* é armazenar transações. Uma transação consiste na movimentação de um ativo financeiro digital de um participante para outro. Esta movimentação implica uma alteração da sequência de blocos que será discutida mais à frente. Estas alterações são feitas seguindo um protocolo de cinco etapas (Figura 22): a etapa de transação, construção do bloco, verificação, função de dispersão e execução.

- Etapa de transação: Durante a etapa de transação há a iniciação de uma transação por um dos dois participantes envolvidos. Esta transação envolve a troca de dados digitais, o que pode inclui a representação digital de ativos físicos.
- Etapa de construção do bloco: Após a iniciação da transação esta encontra-se no estado pendente. De seguida é colocada num bloco com outras transações. Este bloco é um candidato a entrar para o registo imutável.
- Etapa de verificação: Para entrar no registo imutável o bloco deve ser verificado. Esta verificação é dependente da tipologia da rede, discutida na subsecção 4.4.2 . A rede deve verificar se o bloco é válido para entrar no registo, sendo esta verificação normalmente feita através de um protocolo de consenso, para que de forma distribuída a rede aceite este novo bloco.
- Etapa de função de dispersão: Antes de entrar para o registo, cada bloco que foi verificado, é marcado com a hora atual e é associado o resultado da função de dispersão – o *hash* – do bloco anterior. Este passo é necessário para criar um elo de ligação entre o novo bloco e o anterior. O *hash* do bloco reflete todas as transações armazenadas em um determinado bloco e o *hash* do bloco anterior. É esta ligação por *hashs* que dá a garantia de imutabilidade ao sistema (o custo associado de uma mudança é extremamente elevada e aumenta com o tamanho da sequência e a profundidade do bloco no *ledger*).
- Etapa de execução: Neste momento já existe um novo bloco com a transação criada na primeira fase armazenada de forma imutável na sequência de blocos do registo. É considerado que houve a execução da transação e ambas as partes podem considerar que os ativos foram movimentados entre as entidades.

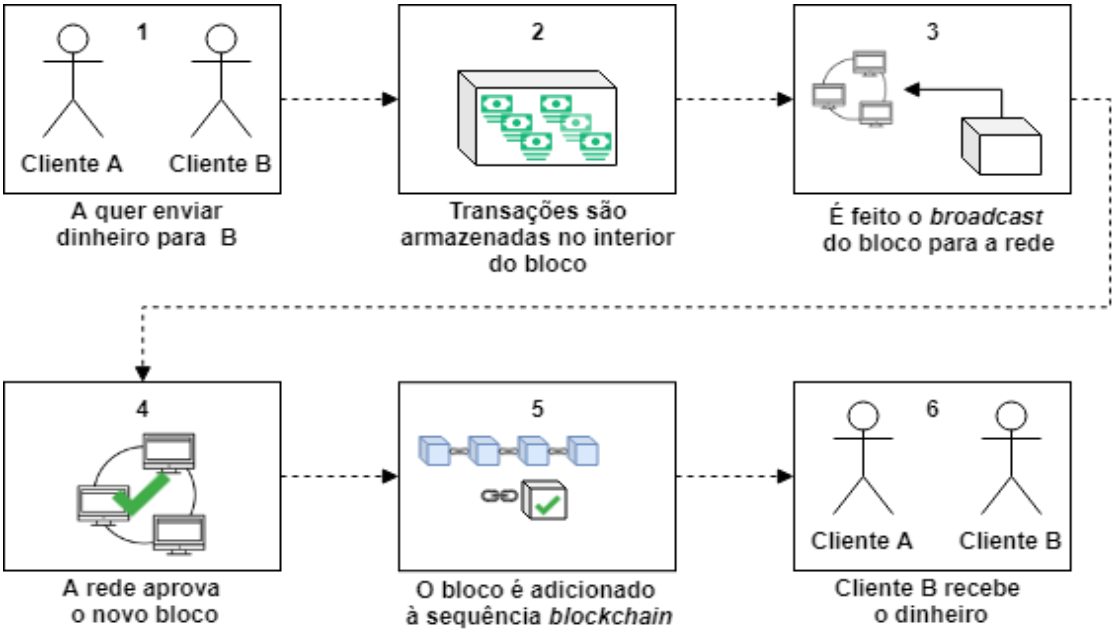


*Figura 22 - Funcionamento da rede blockchain*

### 4.4.2 Tipologia

Os *ledgers* distribuídos podem pertencer a categorias diferentes de acordo com o tipo de permissão associada ao acesso de registos e a permissão associada a entrada de novos nós na rede. Existem assim três combinações diferentes para a rede (50).

***Privada***

Rede privada controlada por apenas uma organização onde as permissões de escrita estão restritas apenas a uma entidade. Permissões de leitura podem ser públicas ou restritas a determinados participantes.

***Consórcio***

Rede controlada por um consórcio de organizações. Todas as entidades são conhecidas e existe confiança entre estas. Permite-se assim neste tipo de redes a escrita de registos por todos os membros. Todas as organizações participam na validação dos registos através do protocolo de consenso. As permissões de leitura podem ser públicas ou restrita a alguns membros.

***Pública***

Qualquer entidade pode aceder à rede, sendo esta rede hospedada em servidores públicos. Estas entidades normalmente podem ler registos existentes nos blocos e possivelmente escrever novos desde que participem no protocolo consenso específico para este tipo de rede.

### 4.4.3 Componentes

***Rede***

Redes *Peer-to-Peer* (P2P) (51) é uma rede de nós que comunicam diretamente entre si através da Internet sem necessitar a existência de entidades intermediárias. Desta forma é possível partilhar os blocos entre os vários intervenientes da rede *blockchain.* Normalmente todos os intervenientes da rede devem possuir toda a sequência de blocos gerada, e distribuir para os restantes os novos blocos que irão ser inseridos. Isto garante a segurança do sistema, sendo que se alguma entidade maliciosa desejar modificar a sequência, é preciso alterar todas as sequências em pelo menos metade dos nós. Esta vertente de segurança é explicada mais à frente na subsecção 4.4.4 sobre as características dos *ledgers* distribuídos.

***Bloco***

Um bloco pode ser visto com uma página no *ledger*. Um bloco pode ser dividido em duas partes como pode-se ver na Figura 23: o cabeçalho e o corpo. O cabeçalho armazena informação sobre a versão do software, o resultado da função de dispersão do bloco anterior, a hora de criação e outras informações. No corpo do bloco estão armazenadas as transações que foram verificadas e estão aptas para entrarem no registo, quando o bloco é inserido na sequência de blocos este confirma que estas transações foram executadas.

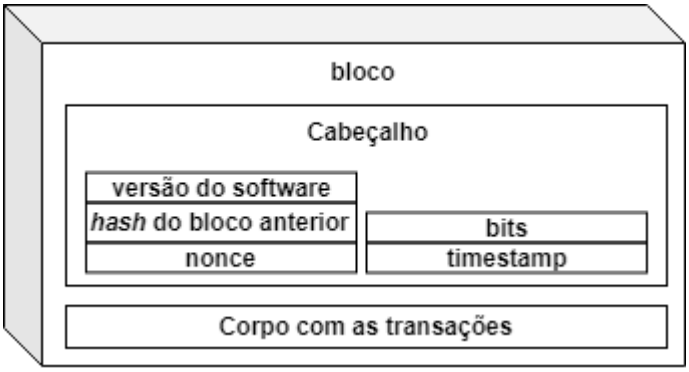


*Figura 23 - Interior do bloco*

***Função de dispersão***

São funções (52) especiais que convertem dados de entrada da função em uma sequência de caracteres, ou *hash*, de tamanho fixo. Estas funções possuem propriedades de segurança importantes e daí o seu papel nesta tecnologia. Estas funções são extremamente importantes pois através da verificação do *hash* gerado é possível garantir a integridade dos dados armazenados em cada bloco

### *Blockchain*

*Blockchain* é o nome dado à estrutura de dados que é gerada aquando da entrada de novos blocos. A sequência é cronológica crescente, pois é garantido que o próximo bloco a entrar no *ledger* foi criado depois da criação do anterior. O elo de ligação entre blocos é feito através das funções de dispersão.

### *Protocolo de consenso*

É o protocolo utilizado na comunicação entre os intervenientes do *blockchain* numa rede P2P distribuída. Este protocolo é utilizado sempre que se pretende dar entrada de um novo bloco na sequência. Tipicamente – a depender do algoritmo de consenso – todos os intervenientes devem chegar a mesma decisão de deixar ou não entrar o novo bloco na sequência.

### *Assinaturas digitais*

Assinatura digital (53) é um processo cujo propósito é autenticar o assinante e garantir que os dados enviados não foram alterados durante a transmissão. Garante-se assim tanto a autenticidade como a integridade dos dados. As assinaturas digitais recorrem à utilização de um par de chaves, uma chave pública e uma privada. A chave privada é utilizada para assinar os dados e a chave pública permite ao público verificar a veracidade dos dados. Esta verificação pode ser referente à integridade dos dados, garantindo que os dados não foram modificados, ou ser referente à autenticidade, garantindo que apenas a entidade que possui a chave privada foi capaz de assinar. Além disto as assinaturas digitais impedem situações de repúdio, ou seja, se o documento foi assinado por uma entidade, apenas esta entidade o poderia ter assinado, pois apenas esta possui a chave privada requerida para o fazer.

### *Smart contracts*

*Smart contracts* (54) são pedaços de código que são executados na *blockchain* de forma independente e que têm como propósito impor regras de negócio sobre as transações. Desta forma garantindo a segurança e credibilidade das transações sem necessitar a presença de entidades intermédias. É inegável que a ausência destas entidades faz com que as transações sejam rápidas e com baixos custos para os intervenientes.

Os *smart contracts* são poderosos porque possuem lógica de programação que não pode ser alterada após criação, o que implica que o *smart contract* é executado tal e qual a vontade do desenvolvedor. São programas simples do tipo *if then – else that* onde caso todas as condições no contrato forem aceites por todos os intervenientes, a transação é considerada válida.

### 4.4.4 *Características*

***Reconciliação através da criptografia***

Instituições tais como empresas e governos atualmente comunicam entre si através do envio de mensagens com os detalhes das transações. Na receção das mensagens cada instituição atualiza o seu registo interno. Não há uma forma fácil de verificar que todas as instituições estão a armazenar os mesmos detalhes e na mesma ordem que todos os restantes.

Com a utilização de um *ledger* distribuído entre as instituições é garantido, pela natureza da rede, que todos os nós recebem a mesma mensagem e é garantido que a sequência de mensagens é igual em todos os nós e que todos estes aceitam os mesmos detalhes.

***Replicação dos dados***

Todos os nós da rede possuem uma cópia, ou parte, do *ledger* do sistema. Desta forma evitam-se pontos únicos de falha do sistema e resolve-se o problema habitual dos sistemas informáticos que é a replicação dos dados que criam custos e complexidade para as instituições.

Quanto maior for a rede, mais replicados os dados estarão pelo sistema, garantindo assim a confiança de que os dados não serão perdidos.

***Confiança através da descentralização***

Um *ledger* distribuído é uma base de dados descentralizada e, portanto, não existe uma entidade central que orquestra como este armazenamento deve ser efetuado. Assim é retirado um ponto único de falha e garante que não há uma entidade com poder suficiente para modificar o sistema, sendo um sistema mais democrático, onde os nós tanto armazenam o registo como participam na aceitação de novos registos.

As entradas de novos registos no sistema devem ser aceites pelos nós do sistema. Esta aceitação deve ser realizada pelos nós de validação existentes no sistema, no caso do *blockchain* normalmente são todos os nós. Isto cria confiança no sistema sendo este o ponto mais forte e o ponto mais fraco da rede distribuída e descentralizada. Caso um agente malicioso queira atacar a rede ele deve dominar pelo menos metade mais um dos nós da rede para que este tenha a capacidade de aceitar os registos maliciosos. Naturalmente a confiança é proporcional ao tamanho da rede, quanto maior a rede mais confiança pode-se ter.

***Custo***

Tipicamente quando os utilizadores pretendem validar uma transação bancária, ou um documento, é necessária uma entidade ou instituição que verifica e válida os dados e tipicamente é necessário pagar os custos associados. Esta é a razão pela qual as pequenas empresas incrementam nos produtos vendidos uma pequena taxa que ira ser utilizada para pagar os custos associados a transação no caso de pagamentos utilizando cartões de crédito e débito. Num *ledger* distribuído não existe esta entidade ou instituição que aplica taxas sobre as transações, assim sendo, as transações na rede não apresentam custos – ou custos muito reduzidos – aos intervenientes.

***Eficiência***

Transações efetuadas através de uma entidade central podem levar até alguns dias para efetivamente efetuarem as devidas movimentações. Isto acontece porque normalmente as entidades financeiras apenas trabalham durante os dias úteis, sendo que 28% da semana é assim perdido. Como a rede *blockchain* se encontra ativa 24 horas por dia todos os dias as transações podem ser efetuadas a qualquer momento, apenas demorando o tempo necessário para criar um novo registo, validar o registo perante todos os nós de validação e integrar o novo bloco na sequência existente.

***Segurança***

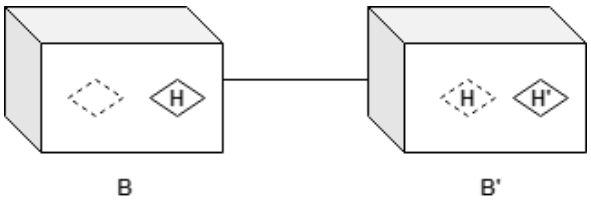


*Figura 24 - Elo de ligação entre blocos através do hash*

Associado a cada bloco B existe um *hash* H. Então pode-se gerar uma sequência se o bloco B', que é o próximo bloco da sequência, armazenar o *hash* H do bloco B. Desta forma como sabemos que o B' armazena o *hash* H, o seu *hash* H' é gerado a partir dos dados armazenados em B' em conjunto com o hash H. Caso os dados armazenados em B se alterem, isto implica que o *hash* H passa a ser diferente e, consequentemente, os dados de B' e o *hash* H' também têm de ser alterados. Desta forma é criada uma sequência de blocos que na prática são muito difíceis de alterar. Caso um bloco seja alterado, todas os blocos da sequência a partir do bloco B têm de ser alterados e é em geral impraticável a computação associada a esta mudança.

Assim, pode-se concluir que os blocos mais antigos são mais resistentes a mudanças. O que quer dizer que existe um determinado nível de desconfiança associado

aos blocos mais recentes que são inseridos no *ledger*. Desta forma, as entidades devem esperar para que sejam inseridos novos blocos após o bloco que estas fizeram transações, apenas por precaução.

## 4.5 Tecnologias de *ledgers* distribuídos

Nesta secção pretende-se apresentar brevemente as tecnologias mais predominantes para o desenvolvimento de *ledgers* distribuídos, que tiram partido da estrutura de dados *blockchain* ou que foram inspirados neste para armazenar os dados.

### 4.5.1 Ethereum

Ethereum (55) é uma plataforma *open-source*, pública e baseada em protocolos *blockhains*. Foi fundada Vitalik Buterin e disponibilizada publicamente em julho de 2015. Permite aos desenvolvedores criar sistemas descentralizados que tiram partido das tecnologias *blockchains*. A plataforma possui a sua própria cripto-moeda, o Ether (56), e uma carteira digital, a *e-Wallet*. A grande vantagem desta plataforma é a disponibilização de criação de *smart contracts* escritos em Solidity (57), uma linguagem criada especificamente para desenvolvimento de *smart contracts* na plataforma Ethereum.

### 4.5.2 Hyperledger Fabric

Hyperledger Fabric (58) é uma plataforma *open-source*, formada através do esforço colaborativo entre múltiplas indústrias relacionadas com o *blockchain*. É hospedada pela fundação Linux (59) que lançou a plataforma em 2016. O Fabric oferece uma plataforma neutra e aberta com suporte para o desenvolvimento em Python (60), JavaScript (61) e GO (62).

### 4.5.3 Corda

Corda (63) é uma plataforma inspirada na *blockchain* e de *smart contracts* direcionada para o sector financeiro. Como plataforma *blockchain*, o Corda permite aos intervenientes transacionar valores diretamente e de forma instantânea. *Smart contracts* permite ao Corda ter acordos mais complexos e de qualquer conteúdo. Pode-se escrever aplicações corda, as CorDapps, em Java ou Kotlin (64).

## 4.6 Solução

Nesta secção é discutida a solução desenhada para o sistema de pagamentos interoperável e, posteriormente, implementada. O Corda foi a plataforma escolhida para o desenvolvimento do sistema. Esta decisão assenta em dois aspetos:

- Linguagem de programação: É possível desenvolver na plataforma Corda em Java ou Kotlin. Havendo familiaridade com ambas as linguagens, sobra mais tempo para aprender sobre a plataforma. Outras opções implicavam despender tempo a aprender uma linguagem nova.
- Sistema financeiro: O Corda é uma plataforma direcionada para sistemas financeiros, disponibilizando um SDK rico em funções que facilitam e aceleram o desenvolvimento de sistemas deste tipo.

### 4.6.1 Visão geral da solução

A presente subsecção pretende apresentar detalhadamente uma solução para o problema descrito na secção 0, a qual passa por desenvolver um sistema que tira partido das vantagens fornecidas pelas tecnologias da plataforma Corda através da criação de um *ledger* distribuído do tipo consórcio.

Mais precisamente, a solução passa por ter uma rede de nós, onde cada nó representa uma entidade de pagamentos, que pode realizar ou solicitar os pagamentos a outras entidades, também nós da rede. Associado a cada nó existe um *ledger* e é neste que são armazenados os dados relativos aos pagamentos.

A Figura 25 mostra uma vista da arquitetura do sistema desenvolvido, o qual utiliza a plataforma Corda.

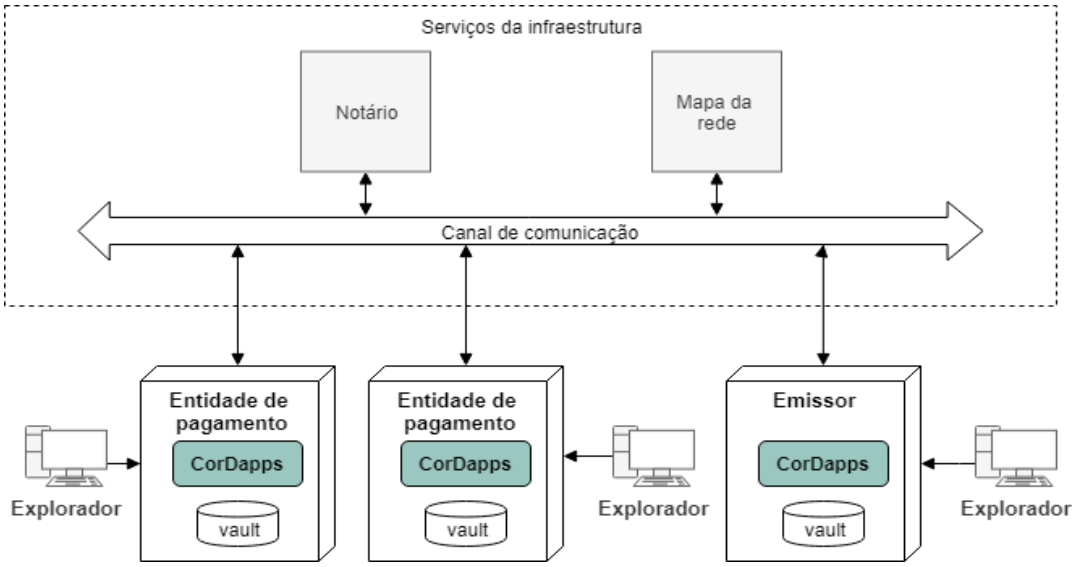


*Figura 25 - Infraestrutura Corda do sistema de pagamentos interoperável*

O explorador representa o cliente de uma entidade de pagamentos. Este é o ator que inicia um pagamento ou que solicita um pagamento. É expectável que este explorador seja um outro sistema de software – interno à entidade de pagamentos – que faz pedidos para o nó a que este encontra associado.

O emissor e cada entidade de pagamento dá origem a um nó do sistema. Em cada nó há dois componentes principais a executar:

- CorDapps: O componente que orquestra a criação de uma nova transação, tem os *smart contracts* associados e SDK necessário para o funcionamento do nó. O comportamento deste componente é determinado pelo código específico da aplicação.
- Vault: Este componente guarda os estados relevantes para o nó. Na plataforma Corda, os dados relevantes que o desenvolvedor deseja armazenar no *ledger* devem ser embrulhados num objeto do tipo estado. Como exemplos de estados temos o pagamento e a bolsa de *tokens*. Mais detalhes sobre os estados são apresentados na subsecção 4.6.4

Normalmente em uma rede *blockchain*, todos os nós participam da validação das transações. Isto porque é necessário provar a exatidão, autorização e unicidade, para que estas possam entrarem no *ledger*. Na plataforma Corda, a exatidão e autorização apenas é verificada pelos intervenientes diretos da transação. Se Alice deseja enviar 10 euros para o Bob, a Alice deve ter meios de autenticar o Bob e vise versa, e apenas Alice e Bob têm interesse – ter um contrato entre os dois – em saber sobre esta transação. Portanto Charlie não deve se envolver na transação. Eis que surge um problema neste tipo de abordagem, a unicidade da transação. Para solucionar a unicidade da transação é necessário que a plataforma forneça mecanismos que garantam que Alice não transaciona os mesmos ativos digitais para Bob e Charlie ao mesmo tempo (problema dos gastos duplos). Este mecanismo é o serviço notário. O notário deve assinar todos os estados, garantindo assim a unicidade da transação e finalização.

O notário armazena internamente um mapa entre a transação e os estados de entrada da transação. Caso existam uma transação que consome, pelo menos um estado já consumido, esta transação e dada como invalida e não única e é rejeitada.

O mapa da rede tem conhecimento de todos os nós da rede e disponibiliza para qualquer nó o meio para que este possa contactar o destino. Cada nó da rede é identificado por uma chave pública. Caso a origem não saiba a chave pública, o nó pode ser identificado através do nome, organização e país do certificado X.509 (65).

Os nós precisam comunicar tanto com os serviços da infraestrutura – o notário, mapa da rede ou outro serviço que seja necessário – como com os outros nós da rede.

Esta comunicação necessita de ser efetuada de forma segura e feita de forma eficiente. Na plataforma Corda esta comunicação é feita através de um canal do tipo *Advanced Message Queuing Protocol* (AMQP) (66) na versão 1.0 sobre a camada de transporte TLS. É assim garantida a segurança e integridade dos dados transmitidos na rede.

O código do componente CorDapp encontra-se dividido em duas camadas lógicas, como pode-se ver na Figura 26:

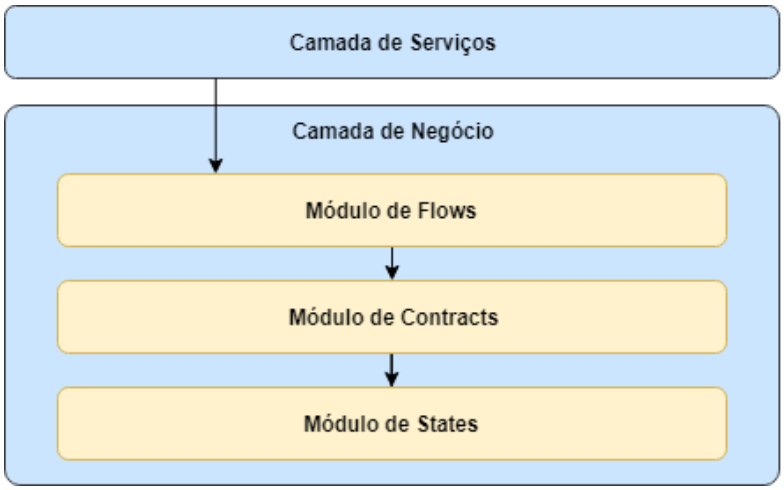


*Figura 26 - Vista de Módulos da CorDapp do sistema de pagamentos interoperável*

- Camada de serviços: Que disponibiliza para o exterior de cada nó uma API REST. Este tipo de API foi escolhido devido a promover a interoperabilidade e tornar fácil a integração com outros componentes. Esta camada será mais detalhada na subsecção 4.6.5
- Camada de negócio: Esta camada é responsável por toda a lógica de negócio do sistema e é constituída por três módulos.
  - O módulo de *flows* é responsável pelas operações do sistema. Para cada operação que o sistema disponibiliza para o exterior, existe um *flow* que serve como ponto de entrada e implementa a operação. Um *flow* é responsável por criar uma transação e associar os estados aos contractos. Estes estados podem ser obtidos do *vault* do sistema ou simplesmente criados durante a execução do *flow*.
  - O módulo de *contracts* tem os *smart contracts* do sistema. Cada transação pode ter associados múltiplos contractos. No caso do sistema de pagamentos, para cada *flow* existe um *contract* que o implementa. Esta decisão serve para simplificar e acelerar o desenvolvimento. Um contracto implementa a lógica de negócio da respetiva operação. Por exemplo, num pagamento, esta lógica passa por o montante ter de ser positivo e a origem ter de ser diferente do destino.
  - O módulo de *states* é responsável pelos estados do sistema. Um estado armazena informação pertinente ao negócio e que de seguida irá ser persistido

no *vault* do nó. No caso do sistema de pagamentos existem apenas dois estados: o pagamento e a bolsa de *tokens*, que armazena informação referente ao pagamento e aos *tokens* da entidade respetivamente.

A Figura 27 mostra os vários passos que constituem a realização de um pagamento entre duas entidades de pagamentos e os vários elementos envolvidos.

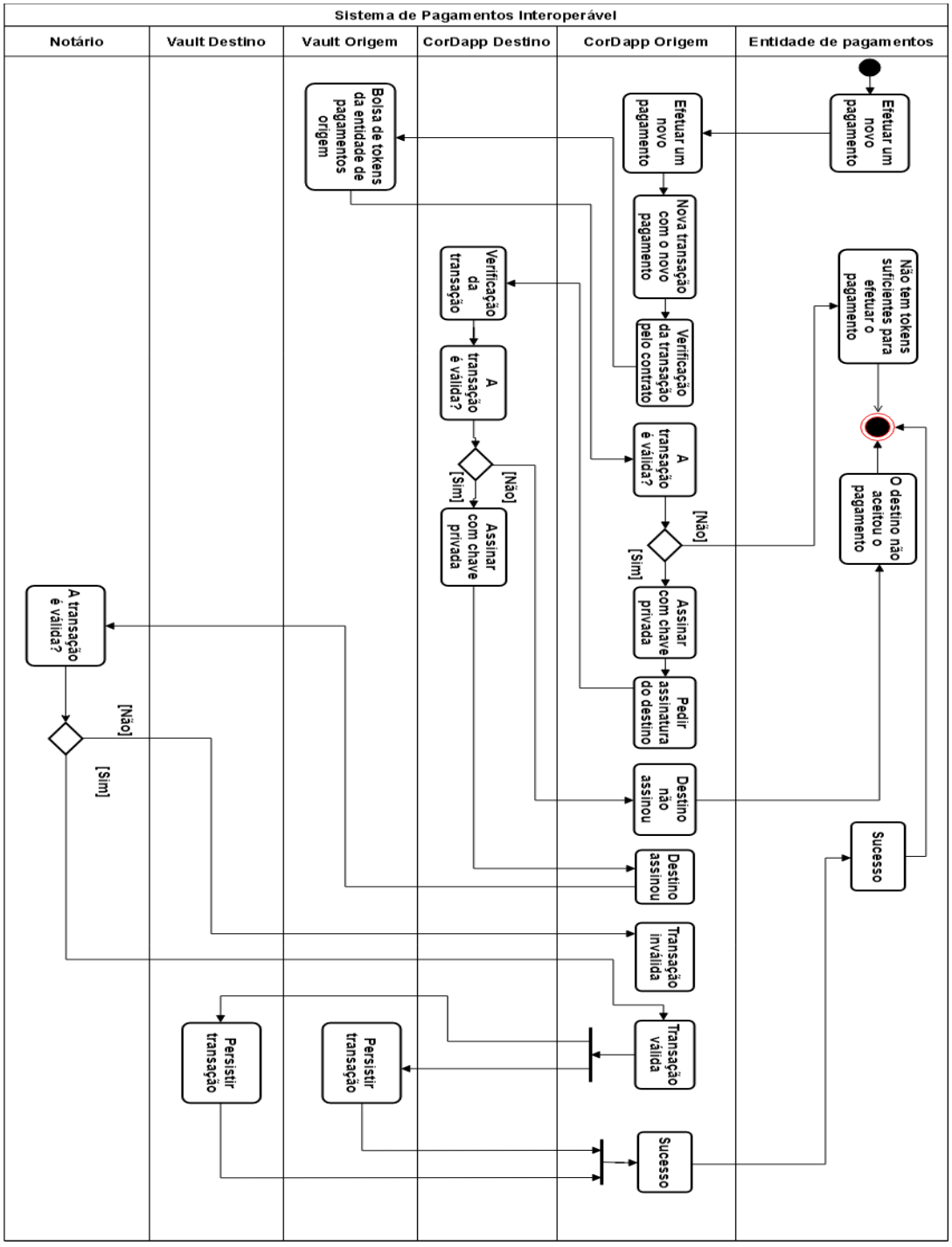


*Figura 27 - Diagrama de atividade do serviço de realizar um novo pagamento*

Qualquer operação no sistema de pagamentos é iniciada na entidade de pagamentos. Esta envia o pedido para o nó, que o representa, para iniciar o pagamento.

O pedido é recebido na camada de serviços – através de uma API REST – que executa o *flow* associado a operação em causa. O *flow* cria uma nova transação e associa os estados relevantes e o contrato da operação.

Após a criação da transação, é necessário verificar se esta é válida. Esta verificação ocorre ao nível do contrato, que possui as regras de negócio associadas a operação em causa. Após a verificação a transação é assinada pelo nó e é enviada para todos os intervenientes que participam de forma ativa na transação. O facto de não enviar para todos os nós da rede, implica que há menos troca de mensagens e consequentemente a verificação demora menos tempo. Isto só é possível porque a rede é do tipo consórcio e há uma grande confiança entre os nós.

Após a recolha das assinaturas de todas as entidades que têm interesse e que participam na transação, esta é enviada para o notário que verifica a transação referente a sua unicidade. E só então a transação é considerada válida e pode entrar para o *ledger*. Após a persistência, o utilizador é notificado do sucesso da transação.

### 4.6.2 Os tokens

Como mencionado anteriormente, o *token* é usado pelo sistema para realizar a transferência de valor entre duas entidades envolvidas numa transação. É a transferência de *tokens* nativos entre duas carteiras que permite ao sistema garantir a finalidade e realizar a reconciliação financeira em tempo real. Daqui resulta a disponibilidade imediata dos *tokens* para poderem ser utilizados em outras transações.

### 4.6.3 A rede

***Nós da rede***

Tal como pode-se ver na Figura 28 a rede do sistema é composta por três nós distintos que desempenham papeis específicos.

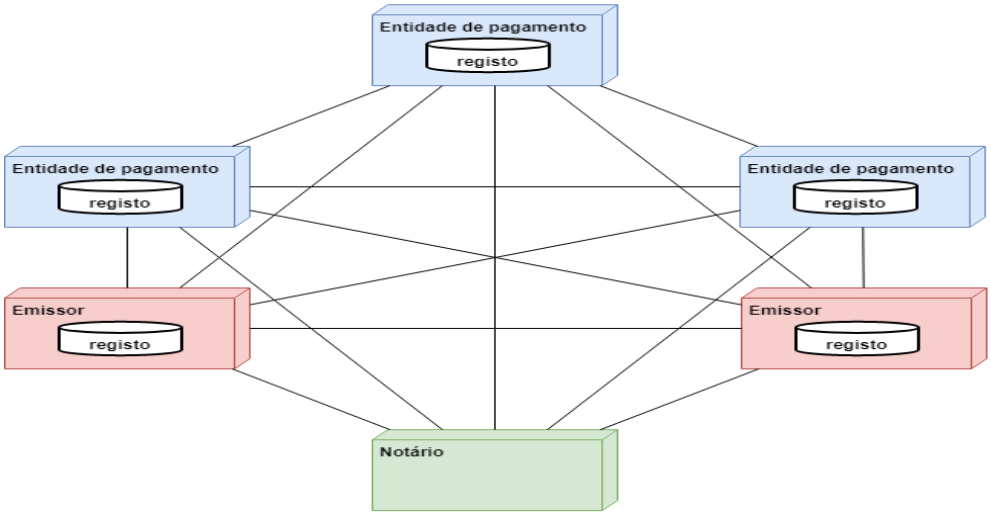


*Figura 28 - A rede do sistema de pagamentos interoperável*

***Nó Notário***

O nó notário é um serviço da plataforma Corda que é responsável por garantir a unicidade, finalidade e validade das transações. Evita-se assim que existam nós a tirar partido do problema dos gastos duplos de ativos digitais – *tokens* – dentro do sistema.

**Emissor**

O nó emissor é responsável apenas pelos serviços de gestão de *tokens*, ou seja, todos os serviços que envolve a criação de bolsa de *tokens*, emissão de novos *tokens* e libertação de *tokens*, devem ser requisitados aos nós do tipo emissor. Para além de fornecer serviços de gestão de *tokens*, o nó emissor deve participar das transações que envolvem a movimentação de *tokens* entre bolsas. Garante-se assim que não há pagamentos de uma entidade para outra de uma quantidade de *tokens* que esta não possui.

**Entidade**

O nó entidade é a virtualização de uma entidade de pagamentos externa ao sistema que é capaz de realizar transações entre a própria e outras entidades

### 4.6.4 O *ledger*

O registo na plataforma CORDA é feito através de uma transação com estados. Os estados são objetos que armazenam os dados que deverão ser persistidos no registo. No sistema em causa apenas existem dois estados: o pagamento e a bolsa de *tokens*.

Antes de introduzir os dados no registo, a plataforma recolhe as assinaturas dos nós responsáveis pela transação e que participam nos estados armazenados no registo. Esta operação é transparente para os desenvolvedores.

***Pagamento***

Estado que representa um pagamento de uma entidade para outra entidade. Armazena informação como a data de criação e data de execução do pagamento, entidades envolvidas, quantidade e moeda que o pagamento, fora do sistema foi realizado; quantidade de *tokens* movimentados entre as bolsas.

***Bolsa de tokens***

Cada nó entidade deve encontrar-se na posse de uma bolsa de *tokens*, esta é utilizada para armazenar o saldo disponível. É esta quantidade de *tokens* que permite com que as entidades realizem transferências e reconciliação entre si.

### 4.6.5 Camada de serviços

Os clientes podem interagir com o sistema de pagamentos de duas formas: através de uma API REST ou através de chamadas do tipo *Remote Procedure Calls* (RPC). A API REST é uma representação de um para um dos métodos possíveis de serem chamados através do RPC. Como o método mais simples de interagir com o sistema é através da API REST, apresenta-se a seguir, os serviços do sistema de pagamentos interoperável.

| ID – Nome do serviço<br>[Verbo HTTP] URI<br>Descrição da funcionalidade do serviço |
|---|
| Tokens |
| 1 – Criar bolsa de tokens<br>[POST] /tokens/bags<br>Responsável por criar uma bolsa de *tokens* para um nó do tipo entidade. A bolsa criada tem saldo igual a zero. A bolsa é persistida no registo imutável. |
| 2 – Emitir tokens<br>[POST] / tokens/issue<br>Serviço recebe identificador de um nó do tipo entidade e um valor positivo. É |

| |
|---|
| emitido este valor em *tokens* para a bolsa associada à entidade. |
| 3 – Libertar tokens<br>[POST] /tokens/redeem<br>Serviço recebe identificador de um nó do tipo entidade e um valor positivo. É libertado este valor em *tokens* da bolsa associada à entidade. |
| Pagamentos |
| 4 – Realizar pagamento<br>[POST] /payments/make<br>Recebe uma entidade destino e um montante. Este serviço é responsável por retirar este montante do nó origem e transferir para a entidade destino. Havendo compensação e reconciliação financeira em tempo real. |
| 5 – Solicitar pagamento<br>[POST] /payments/request<br>Recebe uma entidade destino e um montante. Este serviço é responsável por criar uma nova entrada do tipo pagamento no registo imutável em que o estado é SOLICITADO. Não há qualquer movimentação de *tokens* entre bolsas. |
| 6 – Executar pagamento solicitado<br>[POST] /payments/execute<br>Recebe o identificador único de um pagamento solicitado e executa a transferência do montante associado a este pagamento entre as bolsas dos intervenientes. |
| 7 – Histórico de pagamentos<br>[GET] /payments/history<br>Retorna a lista de todos os pagamentos desta entidade. |

*Tabela 4 - Serviços Rest do sistema de pagamentos interoperável*

## 4.7 Conclusão

Este capítulo mostra que é possível criar um sistema de pagamentos entre entidades que possa realizar pagamentos de forma rápida, segura e descentralizado. Este sistema foi desenvolvido numa perspetiva de exploração das novas tecnologias *blockchains*. Onde foi necessário investigar sobre estas tecnologias e verificar se faziam sentido desenvolver um sistema que tira partido destas. Assim foi possível criar um sistema de pagamentos interoperável que tira partido de um ativo interno, os *tokens*, de forma a

evitar tratar de câmbios e troca de moedas. Durante a escrita deste relatório, a empresa acolhedora encontra-se a verificar a viabilidade da implementação deste sistema em ambiente de produção.

Este capítulo apresenta a plataforma Corda como a escolhida para o desenvolvimento, devido ao facto de ter sido desenvolvida a pensar nos sistemas financeiros e por ser uma plataforma onde o desenvolvimento é numa linguagem que o aluno domina. Termina por apresentar uma visão global do sistema, detalhes da implementação e os serviços que este fornece.

# Capítulo 5 Conclusões

Este capítulo apresenta as principais contribuições e acréscimo de valor que este projeto trouxe para a instituição acolhedora e para os seus clientes que possam vir a usufruir do sistema *Digital Wallet* ou do sistema de pagamentos interoperável. Termina por enumerar diferentes aspetos que podem vir a serem melhorados ou desenvolvidos no futuro.

## 5.1 Trabalho desenvolvido

Durante a execução deste projeto de mestrado na Innovation Makers procurou-se encontrar soluções para dois grandes problemas:

- Levar os meios financeiros para junto da população com menos literacia financeira: A solução que ataca este problema passa pela criação de um núcleo bancário que consegue transacionar ativos financeiros através da utilização dos telemóveis. Esta solução possui uma barreira de entrada de novos clientes bastante baixa, o que permite o acesso rápido e simples. No entanto, para mitigar os riscos associados à facilidade de entrada, os clientes são categorizados em um nível de *know your customer*.
- Transferência de ativos entre entidades de pagamentos que tenham um baixo custo e finalização imediata: Para solucionar este problema foi realizada uma investigação sobre as tecnologias de *ledgers* distribuídos que tiram partido de *blockchain.* Estas tecnologias permitem criar redes de entidades descentralizadas e distribuídas. A descentralização permite a retirada da entidade central e assim diminuir os custos associados à transação de ativos. Com a remoção da entidade central a distribuição da rede permite que esta chegue a um consenso sobre os dados que devem entrar no *ledger* distribuído entre os vários nós.

O *núcleo-bancário digital wallet* contribuiu para o enriquecimento dos serviços financeiros que a empresa acolhedora fornece aos seus clientes do sector financeiro. Oferecendo um novo método de transacionar dinheiro através da filosofia de carteiras digitais implementada no núcleo. Durante a execução deste trabalho foi possível ao aluno exercitar conhecimentos previamente adquiridos durante o percurso académico e

adquirir novos conhecimentos, nomeadamente na área de desenvolvimento de aplicações de serviços, como é o caso do *núcleo-bancário*.

O sistema interoperável de pagamentos, desenvolvido em contexto puramente de experimental e de exploração, permitiu verificar que ao tirar partido das tecnologias de *blockchain* é possível desenvolver soluções inovadoras e que podem revolucionar os meios tradicionais de transacionar e armazenar ativos digitais. Durante o processo de investigação foi possível ao aluno aprender sobre as tecnologias de *ledgers* distribuídos que se apresentam como uma forma de realizar transações rápidas e descentralizadas, que tiram partido de tecnologias subjacentes que permitem a segurança e integridade dos dados.

Além de competências técnicas, foram adquiridas competências pessoais, beneficiando muito do facto de estar integrado dentro de uma equipa. Em primeiro lugar, humildade para aprender com os colegas de equipa com experiência no desenvolvimento de aplicações comerciais para o sector financeiro. Em segundo lugar, competências a nível de comunicação de modo a alcançar consenso sobre o trabalho de integração com as outras equipas – que se encontram a manter os componentes que já existiam – que fazem parte do sistema *Digital Wallet*.

## 5.2 Trabalho futuro

Nesta secção pretende-se apresentar as considerações e sugestões de melhoramento para os sistemas desenvolvidos durante este projeto de mestrado. É dada especial ênfase ao sistema *Digital Wallet*, devido às suas características e por ser um produto que pertence ao leque de serviços fornecidos aos clientes da empresa acolhedora.

A execução do sistema *Digital Wallet* introduziu novos componentes neste sistema, o *núcleo-bancário*. Deve haver a manutenção destes componentes pelas equipas de desenvolvimento. As introduções de novas funcionalidades devem seguir a *wiki* no sítio interno da instituição acolhedora.

Existem vários aspetos que podem ser melhorados ou endereçados em trabalho futuro, nomeadamente:

- *Núcleo-bancário digital wallet*
  - Introdução de novos idiomas na base de dados além do português – os clientes da instituição financeira para qual este projeto se destina falam português, além de ser o idioma oficial do país onde a instituição financeira encontra-se presente – para as mensagens de suporte do sistema.
  - Implementação das funcionalidades relacionadas com os agentes, as quais não estavam planeadas serem desenvolvidas durante este projeto de mestrado.

  - Suporte ao núcleo em ambiente de produção, com as devidas correções seguindo o mesmo padrão utilizado para desenvolver este projeto. Ou seja, corrigir em desenvolvimento, testes em desenvolvimento, passagem para qualidade, testes em qualidade e termina com passagem para produção.
- Sistema de pagamentos interoperável
  - Implementar a funcionalidade de expirar pagamentos após um intervalo de tempo, que não foi possível implementar durante a execução deste projeto de mestrado.
  - Implementar um mecanismo de taxas sobre as transações realizadas dentro da rede, de forma a gerar receitas.
  - Analisar se faz sentido durante o *deploy* de uma rede nova, criar as bolsas de *tokens* das entidades de forma automática, caso esta ainda não tenha. Desta forma retirar este serviço prestado pelo nó emissor.

# Capítulo 6 Bibliografia


1. Observatório da Inclusão Financeira 2015. *Delloite.* [Online] [Cited: Julho 29, 2019.] https://www2.deloitte.com/ao/pt/pages/financial-services/articles/banca-em-analise-observatorio-de-inclusao-financeira.html.

2. Parte do PIB circula fora do sistema financeiro. *Jornal de Angola.* [Online] Março 22, 2017. [Cited: Julho 05, 2019.] http://jornaldeangola.sapo.ao/economia/parte_do_pib_circula_fora_do_sistema_financeiro.

3. Digital Wallet. *Wikipedia.* [Online] [Cited: Julho 5, 2019.] https://en.wikipedia.org/wiki/Digital_wallet.

4. Innovation Makers. *Innovation Makers.* [Online] [Cited: Julho 05, 2019.] https://inm.pt/.

5. O retrato da desigualdade em Angola. *sic Notícias.* [Online] Agosto 23, 2017. [Cited: Julho 5, 2019.] https://sicnoticias.pt/mundo/2017-08-23-O-retrato-da-desigualdade-em-Angola.

6. *Distributed Ledger Technology: beyond block chain.* [https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/492972/gs-16-1-distributed-ledger-technology.pdf] UK Government : Office for Science, Office for Science, 2016.

7. BlockChain: Developer's Guide & Use Cases. *Tech Quark.* [Online] Junho 2017. [Cited: Julho 05, 2019.] https://www.techquark.com/2017/06/blockchain-developers-guide-use-cases.html.

8. Central Bank. *Investopedia.* [Online] Maio 21, 2019. [Cited: Julho 5, 2019.] https://www.investopedia.com/terms/c/centralbank.asp.

9. Android Apps on Google Play. *Goole Play.* [Online] [Cited: Julho 5, 2019.] https://play.google.com/store/apps.

10. App Store - Apple. *Apple.* [Online] [Cited: Julho 5, 2019.] https://www.apple.com/ios/app-store/.

11. Android | The World's Most Popular Mobile Platform. *Android.* [Online] [Cited: Julho 5, 2019.] https://www.android.com/.

12. iOS. *Wikipedia.* [Online] https://en.wikipedia.org/wiki/IOS.

13. Communications, GSM Global System for Mobile. *Digital cellular telecommunications system; Unstructured Supplementary Service Data (USSD).* 2000.

14. RFC 2818 - HTTP over TLS. *IETF Tools.* [Online] [Cited: Outubro 5, 2019.] https://tools.ietf.org/html/rfc2818.

15. Know Your Customer. *Wikipedia.* [Online] [Cited: Julho 5, 2019.] https://en.wikipedia.org/wiki/Know_your_customer.

16. *ISO/IEC 18092:2004 : Information technology — Telecommunications and information exchange between systems — Near Field Communication — Interface and Protocol (NFCIP-1).* 2004.

17. Dotcom Bubble Definition. *Investopedia.* [Online] Junho 25, 2019. [Cited: Julho 5, 2019.] https://www.investopedia.com/terms/d/dotcom-bubble.asp.

18. Mobile Wallet. *Wizzit International* . [Online] [Cited: Julho 5, 2019.] https://wizzit.com/mobile-wallet/.

19. Nigeria, m Payment Services. *EzPay Africa.* [Online] [Cited: Julho 5, 2019.] http://ezpayafrica.com/Services.htm.

20. An Interactive Voice Response (IVR) Control Package for the Media Control Channel Framework. *IEFT Tools.* [Online] [Cited: Outubro 5, 2019.] https://tools.ietf.org/html/rfc6231.

21. *ISO/IEC 7816-2:2007 : Identification cards — Integrated circuit cards — Part 2: Cards with contacts — Dimensions and location of the contacts.* 2007.

22. Mpesa Home Page. *Vodafone m-pesa.* [Online] [Cited: Julho 5, 2019.] https://www.mpesa.in/portal/customer/WelcomePage.jsp.

23. Meo Wallet. *wallet.* [Online] [Cited: Julho 5, 2019.] https://wallet.pt/.

24. *JSA - JIS X 0510 : Information technology - Automatic identification and data capture techniques - QR Code bar code symbology specification.* 2018.

25. Google Pay: Pay for whatever, whenever. *G Pay.* [Online] [Cited: Julho 5, 2019.] https://pay.google.com/about/.

26. About | Google. *Google.* [Online] [Cited: Julho 5, 2019.] https://about.google/.

27. MasterCard Contacteless Global. *Master Card.* [Online] [Cited: Julho 5, 2019.] http://www.mastercard.com/contactless/index.html.

28. MB Way Home Page. *MB Way.* [Online] [Cited: Julho 5, 2019.] https://www.mbway.pt/#o-que-e.

29. Empresa - SIBS Site. *SIBS Site.* [Online] [Cited: Julho 5, 2019.] https://www.sibs.com/empresa/.

30. Automated Teller Machine. *Wikipedia.* [Online] [Cited: Julho 5, 2019.] https://en.wikipedia.org/wiki/Automated_teller_machine.

31. Incremental Process Models. [book auth.] Roger Pressman. *Software Engineering: A Practitioner's Approach.* Boston : McGraw-Hill Education, 2009.

32. Fusion Middleware Concepts Guide. *Oracle docs.* [Online] [Cited: Outubro 5, 2019.] https://docs.oracle.com/cd/E21764_01/core.1111/e10103/intro.htm#ASCON110.

33. Overview - Redmine. *Redmine.* [Online] [Cited: Julho 5, 2019.] https://www.redmine.org/.

34. Java Programming Language. *Oracle Docs.* [Online] [Cited: Julho 8, 2019.] https://docs.oracle.com/javase/8/docs/technotes/guides/language/index.html.

35. Java Plataform, Enterpreise Edition ( Java EE ). *Oracle.* [Online] [Cited: Julho 8, 2019.] https://www.oracle.com/technetwork/java/javaee/overview/index.html.

36. About WildFly. *Wildfly.* [Online] [Cited: Julho 8, 2019.] https://wildfly.org/about/.

37. SQL Server 2017 on Windows and Linux. *Microsoft.* [Online] [Cited: Julho 8, 2019.] https://www.microsoft.com/en-us/sql-server/sql-server-2017.

38. Introduction to the Java Persistence API. *Oracle Docs.* [Online] [Cited: Julho 8, 2019.] https://docs.oracle.com/javaee/6/tutorial/doc/bnbpz.html.

39. The Java Persistence Query Language - The Java EE. *Oracle Docs.* [Online] [Cited: Julho 8, 2019.] https://docs.oracle.com/javaee/6/tutorial/doc/bnbtg.html.

40. *ISO/IEC 9075-1:1999 : Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework).* 1999.

41. Fielding Dissertation Chapter 5 - Representational State Transfer (REST). *Donald Bren School of Information & Computer Science.* [Online] [Cited: Julho 8, 2019.] https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.

42. HTTP/1.1: Method Definitions. *W3.* [Online] [Cited: Julho 8, 2019.] https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html.

43. SOAP Version 1.2 Part 1: MEssaging Framework (Second Edition). *W3.* [Online] [Cited: Julho 8, 2019.] https://www.w3.org/TR/soap12/.

44. RPC: Remote Procedure Call Protocol Specification. *IETF Tools.* [Online] [Cited: Outubro 2, 2019.] https://tools.ietf.org/html/rfc1057.

45. Data Mapper. *Martin Fowler.* [Online] [Cited: Julho 8, 2019.] https://martinfowler.com/eaaCatalog/dataMapper.html.

46. RFC 5246 - The Transport Layer Security (TLS) Protocol. *IETF Tools.* [Online] [Cited: Outubro 2, 2019.] https://tools.ietf.org/html/rfc5246.

47. JUnit 4 About. *JUnit.* [Online] [Cited: Julho 8, 2019.] https://junit.org/junit4/.

48. Mockito framework. *Mockito.* [Online] [Cited: Julho 8, 2019.] https://site.mockito.org/.

49. Apache JMeter™. *Apache JMeter.* [Online] [Cited: Julho 8, 2019.] https://jmeter.apache.org/.

50. Types of Blockchain & Distributed Ledger Technologies in General. *BlockchainHub.* [Online] [Cited: Julho 9, 2019.] https://blockchainhub.net/blockchains-and-distributed-ledger-technologies-in-general/.

51. Peer-to-Peer Networks. [book auth.] Andrew S. Tanenbaum and David J. Wetherall. *Computer Networks (5th Edition).* s.l. : Pearson, 2010.

52. Brown, William Stallings and Lawrie. Secure Hash Function. *Computer Security Principles and Practice.* s.l. : Pearson, 2014.

53. Goldreich, Oded. Foundations of Cryptography I: Basic Tools. Cambridge : Cambridge University Press, 2001.

54. Smart Contracts Definition. *Investopedia.* [Online] [Cited: Julho 8, 2019.] https://www.investopedia.com/terms/s/smart-contracts.asp.

55. Ethereum Beginners. *Ethereum.* [Online] [Cited: Julho 8, 2019.] https://www.ethereum.org/beginners/.

56. What is ETH, and how do i get it? *Ethereum.* [Online] [Cited: Julho 8, 2019.] https://www.ethereum.org/use/#_2-what-is-eth-and-how-do-i-get-it.

57. Solidity - Solidity 0.5.12 documentation. *Solidity.* [Online] [Cited: Outubro 2, 2019.] https://solidity.readthedocs.io/en/v0.5.12/.

58. Hyperledger Fabric - Hyperledger. *Hyperledger Fabric.* [Online] [Cited: Julho 8, 2019.] https://www.hyperledger.org/projects/fabric.

59. The Linux Foundation - Supporting Open Source Ecosystems. *The Linux Foundation.* [Online] [Cited: Julho 8, 2019.] https://www.linuxfoundation.org/.

60. Welcome to Python. *Python.* [Online] [Cited: Julho 8, 2019.] https://www.python.org/.

61. JavaScript. *JavaScript.* [Online] [Cited: Julho 8, 2019.] https://www.javascript.com/.

62. The Go Programming Language. *golang.* [Online] [Cited: Julho 8, 2019.] https://golang.org/.

63. An Open Source Blockchain Platform for Businesses. *Corda.* [Online] [Cited: Julho 8, 2019.] https://www.corda.net/.

64. Kotlin Programming Language. *kotlinlang.* [Online] [Cited: Julho 8, 2019.] https://kotlinlang.org/.

65. Management of public keys - X.509. [book auth.] Andrew S. Tanenbaum and David J. Wetherall. *Computer Networks (5th edition).* s.l. : Pearson, 2010.

66. AMQP is the Internet Protocol for Messaging. *AMQP.* [Online] [Cited: Julho 8, 2019.] https://www.amqp.org/about/what.